JAPAN PATENT OFFICE

PATENT LAID-OPEN OFFICIAL GAZETTE

Laid-Open No.
H06-124596

Laid-Open
H.6 (1994) May 6

Application No.: H04-318159

Filed: H.4 (1992) Nov.27

Inventors: Kunihiro Katayama
292 Yoshida-cho, Totsuka-ku, Yokohama-shi, Kanagawa
Microelectronics Equipment Development Research Laboratory, Hitachi Ltd.

Takashi Tsunehiro
292 Yoshida-cho, Totsuka-ku, Yokohama-shi, Kanagawa
Microelectronics Equipment Development Research Laboratory, Hitachi Ltd.

Applicant: 000005108
Hitachi Ltd.
4-6, Kanda Surugadai, Chiyoda-ku, Tokyo

Attorney, Agent: Katsuo Ogawa

(To be continued to the last page)

1. TITLE OF THE INVENTION

A storage System Using Flash Memory

[Summary]

[Objective] This is an invention related to the hardware configuration and the handling of data in a manner that compensate for a limit on erasure count, which is a drawback of flash memory, for the implementation of an auxiliary memory unit using flash memory.

[Constitution] The interior of the flash memory 1 is divided into sectors, which are units of file data write/erase operations, numbers are assigned to the files that are written into the sectors, tables 5 and 6 that keep track of the numbers are provided, and when a previously written file is to be rewritten, data is written as new data into different sectors; the old data is deleted, and the sector is managed on tables 5 and 6 as a new, writable area.

[Effects of the Invention]
Because files that undergo frequent rewriting, such as file management tables, are not allocated to specific sectors, the location of flash memory erasures does not concentrate on a specific locale, but is dispersed, which prolongs the life of the flash memory and increases reliability of the system as a whole.

2. WHAT IS CLAIMED IS:

[Claim 1] A memory device using flash memory comprising: physical sectors, which are flash memory storage areas with numbers assigned; a means for associating logical sector numbers that are used for writing storage data with the physical sector numbers to which data is actually written; and a control unit that performs controls so that, when writing to the same logical sector number, it writes to a non-associated physical sector based on an erasure count so that the number of write operations performed will not be concentrated on a small number of physical sectors.

[Claim 2] In the memory device of Claim 1, a memory device using flash memory comprising: physical sector reference means wherein the logical sector numbers of the data written in the physical sector areas, which correspond to the physical sector numbers assigned to memory areas, are registered; and logical sector reference means wherein into which physical sector the data in a logical sector has been written is registered in terms of physical sector number.

[Claim 3] In the memory device of Claim 1, a memory device using flash memory comprising: erasure count reference means in which an erasure count is recorded for each smallest unit of flash memory erasure.

[Claim 4] In the memory device of Claim 3, a memory device using flash memory wherein erasure count tracking means are provided that check an erasure count in an erasure tracking table each time that data is rewritten such that each time the erasure count reaches a specified integral multiple that is established so that said erasure count is less than a guaranteed memory rewrite count, the erasure count tracking means transfer data from a memory area with a small erasure count and write the data, so as to average out the number of erasures so that a different set of data is stored in memory areas that have a small erasure count.

[Claim 5] In the memory device of Claim 1, a memory device using flash memory comprising: state reference means that reference the state of each memory area of the smallest unit of flash memory erasure.

[Claim 6] In the memory device of Claim 1, a memory device using flash memory wherein a write sector pointer is provided that indicates to which physical sector data is to be written during a write operation on the next logical sector, so that a regularity is conferred on the order in which sectors are written to, in such a way that the write sector pointer is counted up or counted down and the result is used as a memory access address value.

[Claim 7] In the memory device of Claim 2, a memory device using flash memory wherein said physical sector reference means are comprised of non-volatile memory means, and said logical sector reference means are comprised of volatile memory means; wherein, when the system is powered up, the contents of said logical sector reference means are created from the contents of said physical sector reference means that are constructed in said non-volatile memory means, and the results are stored in said volatile memory means.

[Claim 8] In the memory device of Claim 1, a memory device using flash memory comprising: memory means, such that, if the smallest unit of erasure of the flash memory is greater than the memory capacity of a physical sector, the memory area having said smallest unit of erasure is divided into multiple physical sectors, and to erase a physical sector, the memory means temporarily save the data from other physical sectors stored in the memory area within the same smallest unit of erasure.

[Claim 9] In the memory device of Claim 8, a memory device using flash memory wherein, in order to speed up the flash memory writing operation, the memory means that temporarily

store write data are shared with the memory means of Claim 8 that temporarily save data.

[Claim 10] A flash memory chip wherein a memory area is provided, separate from a data storage area, that stores information corresponding to each minimum erasure unit, such that said information can be accessed using the same address as the address for accessing the smallest unit of erasure, and such that data output can be controlled separate from the control of data storage.

[Claim 11] A flash memory chip comprising: a buffer area storing one or more sectors of data separate from the data storage area, such that data can be written into said buffer area from an external source, and the data that is written can be transferred and written to said data storage area.

[Claim 12] A flash memory chip comprising: a buffer area storing one or more sectors of data separate from the data storage area, such that data from said data storage area can be transferred to said buffer area one sector at a time, and the transferred data in said buffer area can be fetched by an external source.

[Claim 13] A flash memory chip comprising: a memory area, separate from the data storage area, that stores information corresponding to each minimum erasure unit, wherein if a faulty memory area arises in the data storage area, that location is registered and corrected.

[Claim 14] An information device wherein the flash memory of Claim 1 is used as an auxiliary memory unit.

[Claim 15] In an information device comprising a CPU, the main memory unit, BIOSROM, which stores initial settings and basic processing programs, and a memory unit using the flash memory of Claim 1 as an auxiliary memory unit, an

information device wherein the programs stored in said BIOSROM include a program that controls an HDD as an auxiliary memory unit, and wherein the memory unit using said flash memory is controlled by said control program.

[Claim 16] In the information device of Claim 15, an information device comprising: test means that determine that the memory unit using the flash memory has reached the limit of its use due to deterioration of the flash memory device, and memory means that store the test results from said test means.

[Claim 17] In the information device of Claim 16, an information device wherein said test means determine that the limit of use is reached when at least one erasure operation on the flash memory, which is a memory medium for the memory unit, does not complete itself within a prescribed time.

[Claim 18] In the information device of Claim 16, an information device wherein said test means add up the memory capacity for which an erasure operation on the flash memory, which is a memory medium for the memory unit, does not complete itself within a prescribed time, and determine that the limit of use is reached when the total exceeds a certain capacity.

[Claim 19] In the information device of Claim 16, an information device wherein said test means record the number of erasure operations on the flash memory, which is a memory medium for the memory unit, and determine that the limit of use is reached when one or more flash memory units exceed a prescribed count.

[Claim 20] In the information device of Claim 16, an information device wherein said test means record the number of erasure operations on the flash memory, which is

a memory medium for the memory unit, and determine that the limit of use is reached when the memory capacity of the flash memory that has exceeded a prescribed count exceeds a prescribed value.

[Claim 21] In the information device comprising an auxiliary memory unit as a memory unit using the flash memory of Claim 1, a central processing unit, and a display unit, an information device comprising: test means that determine that said memory unit has reached the limit of its use due to the deterioration of devices in the flash memory, which is a memory medium; memory means that store the test results from said test means; and means of reporting to said central processing unit when said test means have determined that the limit of use has been reached.

[Claim 22] In the information device of Claim 16, an information device wherein, when said test means have determined that said auxiliary memory unit has reached the limit of its use, said determination is reported to said central processing unit, and a warning is provided to the user by indicating on a display means that the use limit has been reached.

[Claim 23] In the information device comprising an auxiliary memory unit as a memory unit using the flash memory of Claim 1, a central processing unit, and a warning sound generation unit, an information device comprising: test means that determine that said memory unit has reached the limit of its use due to the deterioration of devices in the flash memory, which is a memory medium; and memory means that store the results of testing from said test means; wherein when said test means have determined that the limit of use has been reached, this fact is reported to said central processing unit, and a warning is provided to the user by said warning sound generation unit.

[Claim 24] In the information device of Claim 16, an information device wherein the user can recognize that said memory unit has reached the limit of its use through a maintenance program that reads test results that are determined by said test means and stored in said memory means.

[Claim 25] In the information device of Claim 15, an information device wherein, while the memory unit using the flash memory is performing processing, if the power supply for the information device is shut off, the processing is continued using a battery, which is provided in the memory unit using the flash memory, as a drive power supply.

[Claim 26] In the information device of Claim 15, an information device comprising: a power supply control unit that controls the power supply in the information device, and a transmission means that, if the memory unit using the flash memory is in the midst of processing, transmits the processing status of the memory unit to said power supply control unit; wherein if the central processing unit that regulates data and program's computational processing for the information device stops its operation, said power supply control unit checks the processing status of the memory unit using the flash memory by means of said transmission means, and if the memory unit is in the midst of processing, the power supply control unit continues to supply power to the memory unit using the flash memory, and if the processing has ended and there is no longer a need for the supply of power, the power supply control unit shuts off the supply of power.

[Claim 27] A memory unit using the flash memory comprising: deterioration diagnosis means that diagnose the degree of deterioration of memory cells for each memory area, which is the smallest unit of memory area by which data stored in the flash memory can be erased in batch; deterioration level memory means that store the results of

diagnosis by said deterioration diagnosis means; and memory content replacement means that replace the memory content stored in said memory area based upon the results of diagnosis by said deterioration diagnosis means and the memory content in said deterioration level memory means.

[Claim 28] In the deterioration diagnosis means for the memory unit using the flash memory of Claim 27, a memory unit using the flash memory wherein the deterioration of memory cells is diagnosed based on an erasure count for each memory area, which is the smallest unit of erasure by which data stored in the flash memory is erased in batch.

[Claim 29] In the deterioration diagnosis means for the memory unit using the flash memory of Claim 27, a memory unit using the flash memory wherein the deterioration of memory cells is diagnosed based on the length of time required for the erasure of the memory contents of the flash memory.

[Claim 30] In the deterioration diagnosis means for the memory unit using the flash memory of Claim 28, a memory unit using the flash memory wherein the number of erasure operations performed on each memory area, which is the smallest unit of area for the batch erasure of flash memory.

[Claim 31] In the deterioration diagnosis means for the memory unit using the flash memory of Claim 30, a memory unit using the flash memory wherein the erasure count to be stored is stored by truncating one or more lower bits so as to save the amount of memory which is required to store the extent of deterioration.

[Claim 32] In the deterioration diagnosis means for the memory unit using the flash memory of Claim 27, a memory unit using the flash memory wherein the extent of deterioration to be stored is divided into two or more

levels based on the length of time required for the erasure of the memory contents of the flash memory, a corresponding number is assigned to each level, and the number is stored in memory.

[Claim 33] A memory unit using flash memory comprising: deterioration diagnosis means that diagnose the deterioration of memory cells based on the number of erasure operations performed on each memory area, which is the smallest unit of area of flash memory from which stored data is erased in batch; deterioration level memory means that store said erasure count in memory; and memory content replacement means wherein, if said deterioration diagnosis means have determined that said erasure count has reached a prescribed erasure count, the memory content replacement means identify the memory area which is judged to have a low erasure count because it stores memory contents that resist the occurrence of deterioration, and transfer the contents of said memory area to the memory area that has reached said prescribed erasure count.

[Claim 34] A memory unit using flash memory comprising: deterioration diagnosis means that diagnose the deterioration of memory cells based on the length of time expended on the batch erasure of data stored in the flash memory for each memory area, which is the smallest unit of area of flash memory; deterioration level memory means that divide the degree of deterioration into two or more levels based on the length of time expended on said erasure, assign numbers in correspondence with said levels, and store the results; and memory content replacement means wherein, if said deterioration diagnosis means have determined that deterioration has advanced one or more levels higher than the level of deterioration stored in said deterioration level memory means, the memory content replacement means identify, from said deterioration level memory means, the memory area that stores memory contents that resist the occurrence of rewriting and has a low level

of deterioration, and transfer said memory contents to said memory area with said deterioration level that is one or more steps advanced.

[Claim 35] In the memory unit using the flash memory of Claim 27, a memory unit comprising: translated-address memory means that assigns an address to each memory area and is capable of address translations by storing the addresses of the replaced memory areas so that, when memory contents are replaced, subsequent access to the replaced memory areas is performed correctly; and address translation control means that perform correct access based on the memory contents.

[Claim 36] An auxiliary memory unit using flash memory, wherein, after requesting the writing of data to said auxiliary memory unit, the information device that acts as a host for the auxiliary memory unit using flash memory terminates the output of said write data upon reception of a write-complete signal from said auxiliary memory unit; while said auxiliary memory unit receives the data from the host into a memory unit other than the flash memory, and outputs said write-complete signal to said host information device upon the termination of data to the memory unit other than said flash memory.

[Claim 37] In the auxiliary memory unit using the flash memory of Claim 36, an auxiliary memory unit using flash memory, wherein the write data from the information device that acts as a host for said auxiliary memory unit is stored in a memory unit other than the flash memory, wherein after the write-complete signal is output to said host information device upon completion of the data storage process, the data is transferred from the memory unit other than the flash memory to the flash memory.

[Claim 38] In the auxiliary memory unit using flash memory of Claim 37, an information device comprising: light

emission means in the casings for the information device, wherein, after a write-complete signal is output to the information device acting as a host, the light emission means indicate to the user that the processing is in progress while data is being transferred from a memory unit other than the flash memory to the flash memory.

[Claim 39] In the memory unit of Claim 5, an auxiliary memory unit using flash memory, wherein the status to be referenced is either when said memory area has already been filled with write data or when memory cells have deteriorated and are write-disabled.

[Claim 40] A method for storing data to flash memory, wherein physical sectors with numbers assigned are provided in the memory area of the flash memory; logical sectors with virtual numbers assigned are provided for the writing of memory data; in the process of writing data to said logical sectors, the logical sector numbers are not associated with the numbers for the physical sectors into which data is actually written; even writing to the same logical sector results in writing to different physical sectors; and the writing process is thereby dispersed so that the number of write operations performed on said physical sectors will not be concentrated on a small number of sectors.

[Claim 41] A memory unit using flash memory comprising: flash memory that stores file data while having divided memory areas; erasure tracking means that store cumulative erasure counts on the individual memory areas; and data control means that transfer and write data to memory areas that have a low erasure count compared with a given memory area, and transfer and write data to said memory area with a low erasure count from memory areas that have a high erasure count.

3. DETAILED DESCRIPTION OF THE INVENTION

[0001]
[Scope of Utilization in Industry]
This invention is directed at auxiliary memory units for small mobile information-processing devices; in particular, it relates to semiconductor file storage units using flash memory and information devices that incorporate those units.

[0002]
[Prior Art]
In the prior art for auxiliary memory units for information devices, the magnetic storage unit is the most predominant device. In a magnetic storage unit, the file to be written is divided into storage units called sectors, and they are stored in correspondence with physical locations in the storage medium. In other words, when a file is rewritten, the file is written basically in the same location; to the extent that there is an increase in write data, data is written to new sectors. In contrast to this approach as a different auxiliary memory unit is the optical disc unit. Optical discs that are currently in common use can be written only once, and data cannot be erased from them. Therefore, when a file, already written, needs to be rewritten, no rewriting actually takes place; instead, the file is written to a different area, and the previously written data is nullified so that it will not be read again. Thus, in contrast to the magnetic disk system, the optical disc unit performs the function of an auxiliary memory unit by employing an approach wherein the rewritten data has no relationship whatsoever to the storage location in which the original data was written. In contrast to memory systems that provide the function of auxiliary memory units in which a disc is rotated and a large volume of data is accessed rapidly, semiconductor file storage units that use semiconductor memory as an auxiliary memory unit have received growing interest in recent years. In

particular, non-volatile, electronically rewritable memory (hereinafter referred to as *EEPROM*) is likely to become the mainstream in semiconductor file storage units. Among the technologies that implement this approach is Patent Disclosure H3-25798, which represents a memory unit using EEPROM. The invention circumvents the limitation on the number of rewriting operations that can be supported, which is a drawback of EEPROM, and implements a practical memory unit using EEPROM. In a nutshell, the invention provides a plurality of memory devices, an erasure/rewrite count for each device is recorded and tracked, and when a specified count less than an EEPROM rewrite assurance count is reached, the writing is switched to another set of devices that is provided, so as to protect the stored data.

[0003]

[Problems to Be Solved by the Invention]

In the optical disc unit approach in the prior art as mentioned above, whenever a file is rewritten, a memory area is destroyed, which means that the allocation of an adequate amount of memory areas requires an extremely large-capacity storage medium, which is a problem. Especially, memory units for information devices that involve extensive file rewriting require a large memory area even when the actual storage capacity [occupied by files] may not be large. On the other hand, in the case of magnetic disk systems that are most commonly used as auxiliary memory units, when a file already written is to be rewritten, new data is written to the same area. If this technique is applied to EEPROM, however, the file containing a list of stored files (commonly referred to as the directory file) and the file that provides a reference to the locations of files (the file allocation table) and other files undergo a rewriting every time write access is made to data, with the result that rewriting operations concentrate locally. In EEPROM, which is subject to a limit on the number of write and erasure operations that can be sustained, this type of operation significantly

reduces life. Patent Disclosure H3-25798 applying EEPROM describes an invention wherein the state of deterioration of memory devices is tracked in terms of erasure counts, and before a memory device breaks down, a switchover is made to substitute memory. However, this method requires a memory capacity that is more than double the actual storage capacity. Specifically, the substitute memory is not used at all until such time as the memory that is used first breaks down, and the broken memory, after it has broken down, becomes superfluous. This results in an extremely large amount of waste in terms of physical volume and weight. Moreover, given the fact that normal data rewriting does not occur in the entire memory, the fact that partial deterioration renders the entire memory chip unusable is economically wasteful.

[0004]
[Means of Solving the Problems]
Flash memory, which is a type of EEPROM, permits the electrical erasure of data. Therefore, while being non-volatile memory, it permits the rewriting of data. Further, because the unit of erasure in flash memory is large compared with commonly used EEPROM, flash memory has a simplified cell structure, which permits an increased packing density. Thus, both physically and economically, the flash memory is well-suited for large-capacity auxiliary memory units when compared with other types of memory devices. Therefore, by conducting data storage and rewriting in a manner that minimizes the number of erasure operations in order to circumvent the limit on erasure count, which is a drawback, it becomes possible to use the flash memory as a device that supplants other types of auxiliary memory units. As a means of accomplishing this, similar to optical discs, data is written in such a way that there is little correlation between storage data and storage location, so that when data is to be written, it is appended, and if a previously written file needs to be rewritten, the memory area containing the old file is

nullified and rendered into an erasable area. And with some timing, garbage collection that erases the data in the nullified area is performed. Garbage collection involves the following: if data in a valid area, which need not be erased together with the data in the nullified area because of a large unit of erasure, the garbage collection process transfers the data in the valid area to another storage location, erases the original location, and makes it a new writable area. In addition, an erasure count tracking table that tracks erasure counts is provided, and the erasure count is incremented each time garbage collection is performed. And if a given block reaches a specified erasure count, the data is swapped between this block and a block that does not have a high erasure count.

[0005]

[Operation of the Invention]

According to the means as described above, unlike the optical disc system, memory areas need not be disabled every time data is rewritten, and unneeded data can be erased to make the freed area into a memory area; and unlike the magnetic disk, rewriting operations do not concentrate locally to shorten the life of the flash memory. Even when the writable area shrinks to an extremely small area and the data to be rewritten becomes extremely limited and erased blocks concentrate on specific blocks, by swapping data in a block with a high erasure count with data in a block with a low erasure count, a storage method can be implemented that fixes neither the area in which rewriting operations occur frequently nor the areas in which rewriting operations do not occur frequently, so that the entire memory area can be used uniformly. Consequently, the deterioration of memory is diminished, and the method eliminates the need for the provision of alternate memory.

[0006]

[Embodiments]

Descriptions of embodiments of the present invention follow. First, we describe Embodiment 1 with references to Figs. 1, 2, 3, and 4. Fig. 1 is a hardware configuration diagram for the implementation of Embodiment 1. Fig. 2 shows the internal configuration of the flash memory chip used in this Embodiment. Fig. 3 is a flowchart depicting the main routine for storage data management. Fig. 4 is a flowchart illustrating an erasure tracking routine.

[0007]
First, we describe the operation of the flash memory in reference to Fig. 2. In the figure, Reference Item 11 denotes the entire memory chip; 12, a data write unit; and 13, the smallest unit of data erasure, which we call an erasure block. Flash memory is an electrically erasable PROM; it is a type of EEPROM. Whereas in normal EEPROM, the unit of data writing and the unit of data erasure (rewriting) are the same, in flash memory, the unit of erasure is significantly larger than the unit of rewriting, and to rewrite data that is previously written, a large amount of other data must also be erased. However, the advantage that in flash memory the packing density can be made higher than the normal EEPROM makes flash memory well-suited for large-capacity memory systems. In Fig. 2, the entire memory chip 11 is divided into one or more erasure blocks 13; whereas the unit of writing 12 is one word (a memory word, based on the organization of memory), the erasure block 13 is an area larger than one word. When flash memory is used as an auxiliary memory unit, a 512-byte erasure unit should be employed to match magnetic disk system specifications for ease of use.

[0008]
In the next section, we explain the system configuration and operation of Embodiment 1 of the memory unit using this memory, with references to Figs. 3 and 4. In the explanation that follows, we call the unit of storage of file data a sector. In this Embodiment, 1 sector is equal

to 1 erasure block. In the figures, Reference Item 1 denotes flash memory chips that store file data; 2, an access controller that generates access signals for flash memory 1; 3, a microprocessor that constitutes a flash memory-based external storage system through the manipulation of stored data and status data; 4, program memory that stores control programs that cause the processor 3 to operate; 5, a logical sector table that provides a reference on where in the flash memory 1 the data in a sector (a logical sector) indicated by a logical address is mapped; 6, a physical sector table that provides a reference to the logical sector numbers of file data that is mapped to the physical sectors indicated by physical addresses on the flash memory 1; 7, an erasure count-tracking table that records the cumulative counts of erasure counts of physical sectors; 8, a status table that provides a reference to the status of physical sectors; and 9, a write buffer that temporarily stores write data in order to speed up the data writing process. A description of the operation follows. When there is a read access request from a system (a host system) that makes requests for data access to this system, the processor 8 references the logical sector table 6 and determines the physical sector in which the applicable logical sector is stored, accesses the physical sector, and sends the requested data to the host system. Data write requests from the host system are explained in reference to the flowchart in Fig. 3. The flowchart in Fig. 3 is a flowchart on the main routine for the program that is stored in the program memory 4, the flow of which is explained as follows: First, a write pointer that indicates the sector on which the next write operation is to be performed is set, and whether the sector indicated by the pointer is in a writable state is determined by means of the status table 8 (a). The status table includes a flag that indicates that a sector is unusable due to deterioration because of a high erasure count, and a flag that indicates the data is already written to a physical sector; if these flags are raised and

a given physical sector is write-disabled, the pointer is moved to the next sector (b). If the sector is writable, data is written to that sector (c). If the operation is a rewrite of a logical sector to which data was already written, because the data in the logical sector that was previously written is no longer needed, a physical sector in which unneeded data is written is found from the logical sector table, the physical sector is erased, and simultaneously, the contents of the physical sector table 6 that were written during the previous write operation are erased (d). If the sector is erased, control jumps to the erasure-tracking routine, which will be explained later. Then, the physical sector number indicated by the write pointer is written to the logical sector table 5, and the logical sector number that was written to the location indicated by the write pointer is written to the physical sector table 6 (e). It should be noted that in Step (b), the test whether data is already written may be written to the status table 8; alternatively, this test can be made using the physical sector table 6. For ease of determination, the bits for any unwritten sectors can be set to either all H or all L in the physical sector table 6. In the following, we explain the erasure-tracking routine, mentioned earlier, with reference to Fig. 4. First, the erasure counter in the erasure-tracking table associated with the physical sector from which data was erased is incremented by 1 (a). If the erasure count is less than a specified count, control returns to the main routine; if it is equal to a specified count, data is swapped (b). For data swapping, first the erasure-tracking table for all other sectors is checked, and a sector whose erasure count is the smallest and that has not undergone any data swapping is found (c). The data stored in the sector that was found with a smallest erasure count is written to the sector that previously underwent an erasure operation (d). After the writing process, swapping flags for the two sectors are raised in the status table (e). The swapping flags are intended to prevent the following

possibility: whereas physical sectors with low erasure counts undergo frequent erasure operations by virtue of this routine, if there were no means of indicating this fact, sectors with low erasure counts could conceivably be re-selected, and in such a case, sectors with high erasure counts in which data was swapped could again undergo frequent erasures. When the swapping flag is raised, the contents of the applicable logical sector table and those of the physical sector table are rewritten (f). Upon completion of the above steps, control returns to the main routine. It should be noted that because the sector that was selected because it had the lowest erasure count ends up being erased, the erasure counter in the erasure-tracking table must be incremented. Also, the swapping flag should be cleared when all sector flags have been raised, or the logical test must be reversed, i.e., if the value 1 indicated swapping, now the value 0 should be used to indicate swapping. Further, the specified count in Step (b) should be an integral multiple of a value less than the guaranteed rewrite count for the flash memory. For example, if the guaranteed count is 10000, a multiple of 1000, 2000, or 5000 would be an appropriate choice. This routine ensures that data in a limited block undergoing frequent erasure operations is swapped with data in a block that has a low erasure count so that data from a sector that undergoes relatively few erasure operations is swapped with data in a block undergoing frequent erasure operations, and in this manner, the number of erasure operations can be averaged out. This approach is considered to be highly effective for the storage of data in normal auxiliary memory units. For example, whereas the area in which operating system programs are stored undergoes no rewriting at all, areas containing graphics data or text data, which is applications program data, undergo frequent data rewrite operations. Therefore, if erasure counts are not averaged out, because the memory in the system program area has no way of changing, no deterioration due to an increase in erasure count occurs,

and the memory in any other data areas undergoes frequent erasure operations within a limited memory space, with the result that its erasure count increases rapidly. Thus, the approach described herein can be especially effective under conditions in which there are relatively few available areas.

[0009]
The foregoing is a description of the operation of Embodiment 1. This Embodiment, with the incorporation of a processor, permits detailed control according to the contents of the program memory, it allows the write operations to be sped up through the use of a write buffer, and the provision of a status table provides expansion potential for the recording of sector status information. In addition, the Embodiment provides the benefit of allowing optimal file management through the tracking of erasure counts to prolong the life of flash memory.

[0010]
Next, we describe Embodiment 2 with references to Figs. 5, 6, 7, 8, and 9. Fig. 5 is a hardware configuration diagram for Embodiment 2. Fig. 6 shows the internal memory organization in the flash memory for this Embodiment. Fig. 7 is a flowchart depicting the main routine for the writing of data. Fig. 8 is a flowchart illustrating the cleanup routine that converts sectors containing unneeded data into write-disabled sectors. Fig. 9 is a flowchart for the erasure-tracking routine that keeps track of erasure counts.

[0011]
First, we explain the flash memory chip and the method for its use in this Embodiment with reference to Fig. 6. In the figure, Reference Item 52 denotes a unit of memory area, called a sector, that stores file data. Reference Item 53 denotes the smallest unit of data erasure, called an erasure block, which is comprised of a plurality of

sectors 52. Thus, in contrast to Embodiment 1, the erasure block 53 has a memory capacity different from sector 52. Item 54 denotes the entire memory chip. Although in the figure, this memory chip is comprised of a plurality of erasure blocks, conceivably, there can be one erasure block per chip. In this Embodiment, file data is stored sector by sector, and this Embodiment is applicable to memory chips wherein, when a request is made to rewrite the file data, the file data is erased by simultaneously erasing other sectors as well. Fig. 5 is the hardware configuration of this Embodiment. In the figure, Reference Item 41 denotes flash memory, which is a write data memory area, and each memory area is a memory chip 54; 42, an access controller that accesses the flash memory 41; 43, a processor that manipulates memory data and status data; 44, program memory that stores control programs that enable the processor 3 to run; 45, a physical sector table that provides reference to logical sector numbers that are stored in the physical sector 51 of the flash memory 41; 46, a logical sector table in which physical sector numbers are recorded that provide reference to which physical sector in the flash memory 41 the data in the stored logical sector number is stored; 47, an erasure-tracking table in which erasure counts for blocks are recorded; 48, a status table in which status information on the various blocks is recorded; 49, a write sector count table that provides reference to the number of previously written sectors; 50, a write buffer that temporarily stores write data to speed up write operations; and 51, a cleanup buffer used for the execution of cleanup routines.

[0012]
Next, we describe the operation of this Embodiment. For read access, read access is performed through the referencing of the logical sector table 45 and the physical sector table 46, as in the case of Embodiment 1. On the other hand, if we explain write access with reference to Fig. 7, first, a write pointer is set, and the status table

for the block indicated by the write pointer is referenced (a). Thus, the write pointer in this Embodiment is a block-unit pointer. If the current block is broken down, the pointer is moved to the next block (b). If the current block is not broken down, the write sector count table 49 for the block is referenced (c). If all sectors in the block have already been written to, control jumps to the cleanup routine, which will be explained later. If a sector that has not been written to exists, data is written to that sector (d), and the applicable write sector count table is incremented by 1 (e). Therefore, in Fig. 6 for example, if data written in the immediately preceding operation is written to physical sector 3 of block 1, the next write operation is performed on physical sector 4 of block 1. The actual write access control is performed by the access controller 2. In this case, when all sectors in the block on which write operations are performed have been written to, control jumps to the cleanup routine (f). After the write operations, the sector numbers that were written to the physical sector table and the logical sector table are recorded, respectively (g). If the operation is a rewrite to a logical sector that was previously written to, the logical sector number that was previously written to the physical sector table is erased. This operation is designed to indicate that the data in the physical sector is no longer valid. It should be noted that the operation in Step (f) is intended to reduce the length of required cleanup time; it should not be performed for the purpose of increasing the erasure efficiency for the flash memory. Turning now to the cleanup routine, the cleanup routine is an operation routine that is invoked when the block indicated by the write pointer is already written to all sectors, and the block is write-disabled. The reason that the cleanup routine is needed is that, in the write method that has been described thus far, the rewriting of data in a given file involves the writing to other physical sectors. In other words, although the previously written data is no longer needed before data is rewritten, the

unneeded data remains stored in the memory and should be erased. However, if data is erased every time it becomes unneeded, such an operation can reduce the life of the flash memory with a limited erasure count. Therefore, the erasing is performed by means of a cleanup routine. The cleanup routine is performed when a block is filled with write data. The specific method for cleanup operations is performed according to the flowchart shown in Fig. 8. The following is a description of the parts of the flowchart: The physical sector table for the cleanup block is referenced, and a check is made to determine whether the block contains any erasable sectors, i.e., sectors that are no longer needed because their contents have recently been rewritten to other physical sectors (a). If there is not a single erasable sector, control returns to the main routine; if there is even a single such sector, a cleanup operation is performed. First, the data in the block to be cleaned up is saved to the cleanup buffer 51 (b), and the block to be cleaned up is erased (c). After the erasure operation, control jumps to the erasure-tracking routine of Fig. 9, which will be explained later (d). And, in the cleanup buffer, only those sectors that are still needed are returned to the block to be cleaned up (e). The positions of the sectors when they are returned are such that, if they are returned to their original positions, neither logical sector table 46 nor the physical sector table 46 need to be rewritten. During the returning operation, a method that fills sectors in a block with the smallest sector number first facilitates the writing of the next new sector; however, such a method would require the rewriting of the logical sector table 45 and the physical sector table 46. In any case, the write sector count table must be rewritten to reflect the number of returned sectors (f). Next, we explain the erasure-tracking routine. Fig. 9 is a flowchart illustrating the erasure-tracking routine. Basically, this flowchart is identical to the flowchart that was explained in Embodiment 1, with the exception of the fact that erasure is tracked in units of blocks rather

than in units of sectors. First, a check is made to determine whether the erasure count for the block that underwent an erasure operation has reached a certain count (a). If that count has not been reached, control exits the routine; otherwise, a search is made for the erasure counts for all blocks (b), and a block with the smallest erasure count is found (c). If the block that was cleaned up is not the same as the block with the smallest erasure count, the data in the two blocks is swapped (d), using the cleanup data buffer 51 shown in Fig. 5. And, the swap flags for the swapped blocks are raised (e). The erasure counter is incremented, and both the logical sector table and the physical sector table are rewritten (f). This is the operation of the erasure-tracking routine in this Embodiment. Similar to Embodiment 1, when erasure operations occur frequently in a limited number of blocks, this routine seeks to average out the number of erasure operations by swapping their data with data in the blocks that have received a relatively small number of erasure operations.

[0013]
The foregoing is an explanation of the operation of Embodiment 2. According to this Embodiment, in the cases when the erasure block is too large as a unit of write sectors, the erasure block can be divided for efficient utilization, which is a beneficial effect.

[0014]
Next, we explain Embodiment 3 with references to Figs. 10, 11, and 12. In this Embodiment, as in Embodiment 1, it is assumed that in the memory chip, the erasure blocks and the sectors of Fig. 2 have the same capacity. When the sectors are used, the configuration illustrated in Fig. 10 is assumed. In the figure, Reference Item 91 denotes a block consisting of a plurality of sectors. In this Embodiment, because units of sectors and units of erasure blocks are the same, the block 91 of this Embodiment is comprised of a

plurality of erasure blocks. Other reference numbers, already explained, are the same as those in Figs. 2 and 6. The hardware configuration is shown in Fig. 11. In the figure, Reference Item 101 denotes an erasure-tracking table in which an erasure count for each block 91 is recorded. Although details of this table will be explained later, effectively, the table stores the cumulative results of erasure counts in the block 91. Other reference numbers are the same as those in Figs. 1 and 5. Fig. 12 is a flowchart of the erasure-tracking routine of this Embodiment, and the main routine in this flowchart is the same as in Fig. 3. While the operation of the main routine is the same as Embodiment 1, we now explain, with reference to Fig. 12, the operation that occurs when control jumps from the main routine to the erasure-tracking routine. First, the counter for the erasure-tracking table for the block containing the sectors that are erased is incremented by 1 (a). A test is made to determine whether the erasure count has reached a specified value (b); if it has, the erasure count for the previous block is checked, and the block with the smallest count that has not undergone data swapping is determined (c), and the data is swapped between the two blocks (d). A swapping flag is raised (e), and the contents of the tables are rewritten (f). A feature of this Embodiment is that, in memory that permits erasure operations in units of sectors, erasure tracking is performed on a plurality of sectors, thereby simplifying the erasure tracking process, which can yield the advantages of a saving in wait time in the rewriting of large files and a reduction in the size of the erasure-tracking table. Therefore, this Embodiment is well-suited for large-capacity memory units for which it is difficult to perform erasure tracking on a sector-by-sector basis.

[0015]

We now provide additional explanations on tables, including logical sector tables and physical sector tables, which are configuration elements in the embodiments that have been

described thus far. Given the fact that flash memory is a non-volatile memory, it goes without saying that the system of the present invention does not lose the data in its flash memory when the power supply is stopped when the system is not operating. However, no matter how long data is preserved, if the contents of a table are lost, the system will lose track of what data is stored in what sectors, which can render the data in the flash memory completely meaningless. Therefore, any data stored in a table must also be preserved when the power supply is stopped. However, it is not necessary to preserve all data in the table. For example, the data in the logical sector table can easily be created from the physical sector table. The converse is also true. In other words, it suffices that the data in one of the two tables is preserved. Therefore, the physical sector table is allocated in electrically erasable, writable non-volatile memory (EEPROM), and a logical sector table is created from the logical sector table when the system is started as the supply of the power is commenced. This arrangement permits the use of volatile memory for logical sector tables. Similarly, since the used-sector count table for the various blocks can also be created from the physical sector table, the used-sector count table is created in the volatile memory when the system is started. These tables can also be expanded in the main memory in the main system. Other tables that are used in the system include erasure-tracking tables and status tables, which cannot be created from another table. Because the data in a erasure-tracking table should not be lost, the table is allocated in the EEPROM. The status table is obtained when another write or erasure operation is performed; however, to save the trouble of performing those operations, the status table should be stored in the EEPROM. However, it can also be stored in the volatile memory in order to save memory. As for the medium in which these tables are stored, they can be stored in the same memory in order to reduce the number of required chips. For example, because both the physical

sector table and the erasure count table must be stored in non-volatile memory, they can be stored in the same EEPROM so that only one EEPROM chip will be required. In addition, if the processor is implemented as a single-chip microcomputer, relatively small tables, such as the used-sector count table, should be stored in the RAM core, which is built into the single-chip microcomputer. Fig. 13 shows a configuration diagram for Embodiment 4, which summarizes these arrangements. In the figure, Reference Item 111 denotes a single-chip microcomputer with built-in RAM and ROM; 112, a RAM core in the single-chip microcomputer; 113, a ROM core in the single-chip microcomputer; 114, and an EEPROM chip; 115, either SRAM or DRAM. The other numbers that have already been described are the same as the preceding embodiments. The used-sector count table is stored in the RAM core 112. Microcomputer control programs are stored in the ROM core 113. The physical sector table and erasure-tracking table are stored in the EEPROM 114. A logical sector table is stored in the RAM 115. The free area in the RAM 115 can be used as a cleanup data buffer for the execution of the cleanup routine shown in Fig. 9, and as a write buffer for speeding up the write process. In this manner, according to this Embodiment, tables and buffers can be grouped together to match the features of the memory medium, thereby reducing the number of required chips. By contrast, Fig. 24 is an embodiment wherein the system is configured by omitting EEPROM. This configuration can be implemented by storing the non-volatile table data, explained in Embodiment 4, in the flash memory 1. In the figure, Reference Item 116 denotes a table storage area that is provided in the flash memory 1. However, because flash memory is not well-suited for the updating of small-unit data, such as table data, table data is stored in the flash memory 1 immediately before the power is turned off; in the normal usage state, such table data is stored in volatile DRAM or SRAM 115. In other words, when the power for the system under the present invention is shut off, the required data from the contents

of RAM 115 is transferred to the flash memory 1, then the power is shut off, and when the power is turned on again, the data is loaded from the flash memory 1 to the RAM 115, and normal operations are commenced. Therefore, the memory area 116 for the storage of tables must always be allocated in the flash memory. This storage location, however, need not be fixed. If the location is not fixed, an area indicating the address at which the table is stored is pre-allocated, and the address at which the table is stored is always recorded, and in this manner the need for searching for table positions at the time the power is turned on can be eliminated.

[0016]
Further, Figs. 14, 15, 16, and 23 show embodiments in the configuration of the flash memory itself. Fig. 14 illustrates a memory area for tables that is provided in the flash memory chip, in units of erasure blocks, separate from the data area. By writing physical sector tables and erasure counts for each block into this memory area, the need for EEPROM can be eliminated. In the figure, Reference Item 131 denotes an erasure block; and 132, a memory area for tables, which is added for each erasure block 131. The data stored in the table 132, corresponding to the erasure block 131, can be accessed using the same address as the erasure block 131, and file data and table data can be output separately through selection by means of signal lines and mode selection. This arrangement ensures that the data area and the table area have the same life, and eliminates the possibility of the flash memory being rendered unusable due to a breakdown in the table when the data area 131 is still usable. Such a tendency grows stronger if the data area and the table area are in proximity to each other. In addition, because a table is constructed for each erasure block, the sharing of address lines can be implemented in a simple manner. Fig. 23 (a) shows an example of memory area 132 for tables. In the figure, Reference Item 133 denotes an entire unit block of

erasure; 134, a file data area in the block; 135, an area that stores logical numbers for the block, with an 8-bit memory capacity; 136, a status table with a 16-bit capacity that indicates the status of the block. The contents of the status table 136 include an erasure count-tracking table, a usage-disabled flag, an already-swapped flag, and a correction flag. Any bits that are not used in the status table 136 can also be used to store block numbers. Further, Fig. 23 (b) is an example of table area 132 that is provided with an error correction area, and Reference Item 137 is an area that indicates error correction spots. The memory capacity of this area is set according to the correction capacity of the system. For example, if the error correction area is 8-bit for a memory chip wherein a word is organized in terms of 16 bits per word and has a 512B erasure block organization, the correction capacity is one word. In other words, the number of a word containing bad bits is written in this area, and correction data can be written in other areas. In this manner, correction can be implemented during the read process by replacing the data in the affected word. The correction data area can be provided in the table or grouped in other memory areas. In the former case, all data correction can be handled within a table. However, because a correction data area must be provided for each block, this approach results in a large redundant area in the entire chip. The latter case requires a means for indicating the area in which correction data is written, and because it requires access to other memory areas, this approach results in an increase in access time; however, the redundant area can be set either large or small, depending upon the amount of correction data involved. It should be noted that the area into which correction data is written can be specified by using any excess bits in the status table area 136. In the following, we explain an example of a correction data area with reference to Fig. 15. In Fig. 15, separate from a data memory area, an EEPROM-type memory area 138 that can be written to in small units, such as in one byte or one

word, is provided. Corrections can be implemented by using this area as a correction table, wherein correction data is written into the table in advance, and when data from a location to be corrected is read, the data from a specified location is read and used for replacement. Fig. 15 can be thought of as an embodiment that seeks similar effects by organizing tables that are not in the vicinity of erasure blocks for the corresponding data area, but by organizing them as a group. In this case, the constituent elements in the figure are similar to those in Fig. 14, wherein the memory cells are organized as a group in the chip, which produces the benefit of a simplified memory cell organization.

[0017]
Fig. 16 is a flash memory chip with a different configuration. In this configuration, a buffer area 142, separate from the data memory area 141, which temporarily stores data, is provided in the memory chip. This part can be volatile memory. Reference Item 143 denotes an address counter that counts up based on clock input. During write access, the write data is written into the buffer area 142, and when an address is input, a plurality of data items is at once transferred to the data area 141 so that they can be written. The use of this type of memory eliminates the need for an external write butter for speeding up the write process. Conversely, during a read operation, addresses can be input from the data area 141, and if a plurality of data items can be transferred to the buffer area 142 in a single operation, read access can also be simplified. In this case, the buffer can be serial access memory, which does not require the input of contiguous addresses. The ease of use can be further improved by providing an internal address counter 143, so that when a clock is input, the internal address counter counts up, and data can be output through access to contiguous areas. It should be noted that the most effective organization of a buffer area is one in which each sector is configured as a unit. The

buffering effect can be improved by organizing the buffer into multiple units, not limited to a single unit per buffer. For example, if a sector is an erasure block, and if one buffer that stores data for one sector is provided, read/ write operation on a sector can be performed at a time. If a plurality of buffers is provided, data write operations encompassing a plurality of sectors can be accepted, and for read operations, read data for a plurality of sectors can also be prepared. The use of this buffer provides the benefit of eliminating the need for an external cleanup buffer.

[0018]
In the following, we explain an embodiment that applies the memory unit using flash memory, described in the foregoing, to an information-processing system. Fig. 17 describes an interface circuit that connects a memory unit using flash memory (hereinafter referred to as a "flash file system") to an information-processing system (hereinafter referred to as a "host"). In the figure, Reference Item 201 denotes an external I/O bus of the host. Standard buses that can be used include ISA, EISA, microchannel, and SCSI buses. Reference Item 202 denotes a bus buffer or a bus controller that is used to convert a signal on a standard bus into that for a dedicated bus. Systems in which this bus is omitted can also be considered. Connected to these components is an external storage unit or an auxiliary memory unit that is provided on the host side for the storage of data that cannot be stored in the host system's own main storage unit or extended storage unit, data that cannot be stored in memory units such as display memory units, or data that needs to be retained after the power is shut off. Common among these units are a floppy disk drive 203 and a hard disk drive 204. In addition, a flash file system 205 is connected. It is not necessary that all of these auxiliary memory units are connected to the host system; they can be selected and connected by the user as appropriate. Attached to the flash file system 205 is an

interface circuit 206. Reference Item 207 denotes interface registers; 208, a command register, which is a register in the group of interface registers; 209, an address decoder circuit for the interface registers; and 210, a command interrupt signal. The other numbers that have been described previously are the same as those that have been explained in the foregoing, with the exception of the fact that the programs that are stored in the program memory 4 include a program that responds to commands, centered on access requests from the host, in addition to file data control and management programs. The host issues commands to the auxiliary memory unit through the host bus 201. This operation is performed by writing a command code into the command register 208, which is a member of the group of interface registers 207. The interface registers 207 include all the registers that a hard disk drive has as interface registers, and they have matching register specifications so that they behave exactly the same as when the host accesses the hard disk. We believe that it would be effective to match this register with interfaces for other auxiliary memory units, such as a floppy disk drive or an optical disc drive. Although it may appear as if interfaces for a plurality of auxiliary memory units are supported simultaneously, and as if the host uses a separate auxiliary memory unit, in actuality, a single flash file system handles the functionality of all these units, which is significantly effective in terms of space savings. Now, when a command is written by the host, the command register 208 issues an interrupt signal 210 to the processor 3. Upon reception of this signal, the processor 3 interprets the command code, and responds to the command request from the host. It should be noted that the interface registers 207 and the commands are all compatible with the hard disk drive; however, due to differences in storage media, some registers or commands are superfluous and others may have different processing functions. For example, although formatting is essential to magnetic disk units, it is not needed in semiconductor disk drives;

therefore, no particular formatting processing is performed, or data is simply rewritten to data with regularity. In the following discussion, read/write operations on file data are performed in the same manner as those that were explained in the preceding embodiments. Although the figure applies to Embodiment 1, it can be applied without modification to other embodiments that have been explained thus far or embodiments that will be explained in what follows.

[0019]

Fig. 18 is an example configuration diagram for a personal computer in which a flash file system is used as an auxiliary memory unit. In the figure, Reference Item 221 denotes the CPU for this information device; 222, a coprocessor; 223, a standard I/O bus that is built into the information-processing system of this Embodiment; 224, a bus unit that constitutes the standard I/O bus 223; 225, a memory control unit that accesses high-speed memory, such as the main memory or expansion memory; 226, the main memory; 227, BIOS ROM that stores the basic control program; 228, a keyboard controller, to which a keyboard is assumed to be connected; 229, a display adapter, to which some display device is assumed to be connected; 230, expansion memory; 231, a parallel port interface to which a printer and other devices are connected; 232, a serial port interface, to which a mouse and RS232C devices are connected; 238, a floppy disk drive; 234, a buffer controller that converts the standard I/O bus 223 into a standard HDD interface; and 235, a flash file system. The internal components of the flash file system are organized as described in the previous embodiments. Reference Item 236 denotes an interface unit that receives and passes data upon reception of a file access request, and this corresponds to the interface registers 207 and the address decoder 209 in Fig. 17; 237, a control unit that performs data management and control functions internal to the flash file system 235, comprised of the processor 3, the program

memory 4, the access controller 2, and the write buffer 9 in Fig. 17; 238, a flash memory array that stores file data; 239, an information table that stores information for the management of data and memory, including all of the tables 5, 6, 7, and 8 in Fig. 17. We now explain the operation of the embodiment. When the power is turned on and the flash file system begins to operate, first, the CPU 221 accesses the BIOS ROM 227 through the standard I/O bus 223, and performs an initial diagnosis and initial setting. Then, the system program is loaded from the auxiliary memory unit to the main memory 226. This embodiment incorporates the flash file system 235 as an auxiliary memory unit, with the condition that the CPU 221 operates by accessing the HDD through the standard I/O bus 223 and by using the HDD controller 234. Therefore, the flash file system 235 supports an interface function that is completely HDD-compatible by means of the internal interface unit 236. Upon completion of the loading of the system program, processing is carried out according to the user's processing requests. The user carries out his tasks by performing input/output processing by using the KBDC228 and the display adapter 229 that are installed on the standard I/O bus 223. And, as necessary, the user uses I/O devices that are connected to the parallel interface 231 and the serial interface 232. If the available main memory capacity on the main memory on the system unit runs out, the main memory is supplemented by using the expansion RAM 230. When reading or writing a file is desired, the user makes a request for access to the auxiliary memory unit, assuming that the HDD is an auxiliary memory unit. Upon reception of that request, the flash file system 236 performs access to the file data. In this embodiment, the flash file system is adopted as if an HDD, currently the most common auxiliary memory unit, is installed on a standard I/O bus in a standard personal computer. Therefore, even when a flash file system is adopted as a new storage medium, the BIOS ROM and the system program can be used without modification from a configuration in which

an HDD is installed. Whereas the above embodiment uses a standard personal computer as an example, information devices with a configuration in which the main memory and expansion memory are installed on the standard I/O bus and a coprocessor, a parallel interface, or a serial interface does not exist are also conceivable; the present invention can also be applied to these configurations.

[0020]
In the following, we explain an embodiment that can handle the occurrence of a malfunction in flash memory, with reference to Fig. 19. As mentioned previously, in principle, flash memory is subject to a limit on the number of times it can be rewritten. As deterioration due to rewriting progresses, the amount of time consumed in erasing or writing data increases, and problems arise in data retention reliability. Therefore, if it is determined that the flash memory has reached a condition where it should not be used due to memory deterioration, the use of its memory area or memory chip should be suspended. As such areas increase and if it is determined that the flash file system as a whole has approached an expiration point, the use of the flash file system should be suspended. Conceivable methods for the detection of deterioration of flash memory include a method based on inadequate erasure, wherein when erasure operations are conducted repeatedly, complete erasure is not accomplished within a specified number of operations or the erasure time is greater than a specified length of time; a method based on inadequate writing, wherein, when write operations are conducted repeatedly, complete writing is not accomplished within a specified number of operations; and a method based on tracking the erasure count, wherein the number of erasures conducted has exceeded a specified count. And when the amount of memory that has become unusable due to deterioration has reached a specified value, the flash file system is judged to have reached the limit of its life. Fig. 19 shows an embodiment that implements a means of

warning the user about this fact. In the figure, Reference Item 240 denotes an interrupt signal that conveys to the host system that the controller inside the flash file system has recognized the limit of use of its own system, and 241 denotes an error report register that indicates that the system has reached its usage limit. Upon reception of the interrupt signal 240 through the HDD buffer 234, the host accesses the register 241 in the flash file system, grasps the situation, and reports to the user as appropriate. Fig. 20 shows software that implements this operation in terms of a flowchart, wherein (1) is a flowchart for a check routine in the flash file system; and (2), a flowchart for a user warning program in the host system. In the following, we describe the operation of the software while explaining the figure. When detecting a memory area that appears to have reached its usage limit, the controller for the flash file system jumps to this routine, and adds the capacity of the expired memory area (a). When a certain usage limit has been reached (b), as an error description, the controller writes a value indicating that the usage limit has been reached to the error report register in its system (c). Then, the controller outputs an interrupt signal to the host (d), and exits from the routine. It should be noted that the sum of capacities in Step (a) must be retained either in the flash memory or in other types of memory. The limit on the remaining capacity, shown in Step (b), must be set according to the particular system, as appropriate. The host system, which receives an interrupt signal according to Step (d), suspends the processing at a suitable stop point, and enters into the routine shown in flowchart (2). First, the host system accesses the error report register in the flash file system to determine the reason for reception of the interrupt (e), and if the interrupt is a usage suspension request because the flash file system has reached the limit of its use (f), the host system prohibits further access to the flash file system (g), and warns the user (h). Conceivable methods of warning include: a visual

method, in which a warning indicator using a display device in the information device is installed; and an audio method that generates a warning sound. In addition, only write access should be prohibited with the read access intact, so that the file data can be backed up. This Embodiment is capable of immediately notifying the user when the flash file system has reached its usage limit, thus enhancing data reliability. In another embodiment, the interrupt signal 240 of Fig. 18 is not provided; instead, on each access, the host system reads the error report register, and executes the routine (2) described in Fig. 20. This Embodiment, while involving no changes to the conventional hardware configuration for the host system, can produce usage limit reports by directly replacing the HDD. In another embodiment, the host system periodically executes a maintenance program, and reads the error report register 241 to check whether the usage limit has been reached and recognizes the usage limit. This embodiment does not require any modification of the BIOS program for the host system or the system programs.

[0021]
In the following, we show an embodiment that handles processing when the power supply is shut off. The flash file system under the present invention is basically an auxiliary memory unit, for which the general configuration is that the power for the flash file system is supplied from the host system. However, because operationally the flash file system runs asynchronously from the host, the flash file system may be running when the host is not running. However, if the user handling the host is unaware of it, conceivably, the user can inadvertently shut off the power for the host, which results in the stoppage of the supply of power to the flash file system when it is still running. Fig. 21 illustrates an embodiment that addresses this situation. Fig. 21 is a system diagram for a configuration that supplies the battery-based backup power so that the flash file system can continue to run even

after the supply of power from the host is shut off. In the figure, Reference Item 251 denotes an information device, such as a personal computer, that acts as a host; 252, a flash file system; 253, a backup battery; and 253, a reverse flow breaker circuit that prevents the power from the battery from flowing back to the host 251 when the power from the host 251 is shut off; 256, a detection circuit that detects the stoppage of power supply from the host 251; 256, a detection signal, which is an output from the detection circuit; 257, a circuit breaker that shuts off the power from the battery 253 to the flash file system 252; and 258, a shutoff signal that trips the circuit breaker 257 because the flash file system 252, which terminated the processing that was running, no longer needs power. When the power is supplied from the host 251, the flash file system runs on the power supply for the host 251. By suitably adjusting the output voltage from the battery 253 and the supply voltage for the host 251, trickle recharging can be effected on the battery 253. However, if the battery 253 is not a secondary battery, flow-prevention diodes need to be added. When the power for the host 251 is shut off and the supply of power to the flash file system 252 stops, the power is supplied from the battery 253, which enables the flash file system to continue to run. When the power supply shutoff detection circuit 266 detects that the power supply has been shut off, the flash file system recognizes this fact by means of the detection signal 256, completes the current processing, and subsequently performs power supply shutoff processing. The power supply shutoff processing at the end uses the shutoff signal 258 to shut off the battery circuit breaker 257 and terminates the operation of the flash file system. The types of reverse-flow shutoff circuits 254 that can be used include a reverse-flow prevention diode. In this Embodiment, the host 251 is completely isolated from the flash file system in terms of signals, the flash file system only performs data exchanges in response to access requests from the host, and any stoppage of operation of

the host can be confined to the host. In other words, the host does not require a modification to its hardware due to the connection of the flash file system.

[0022]
Fig. 22 illustrates another embodiment that handles the shutoff of the power for the host. The example given in Fig. 22, however, involves a method that actually continues to supply power from the power supply for the host. In the figure, Reference Item 258 denotes a power supply line leading from the host 251 to the flash file system 252; and 259, a power supply busy signal that indicates that the flash file system 252 is running and that it requires the supply of power. The power supply circuit for the host 251 does not stop the supply of power as long as the power supply busy signal is active, even when the power supply switch is turned off; it stops the supply of power when all processing by the flash file system has terminated and the power supply busy signal has become inactive. In other words, from the user's perspective, the actual operation is not shut off even when the power switch for the host information device is turned off, and the host waits until the processing by the flash file system 252 terminates. In this embodiment, the host controls the power supply by recognizing the termination of operation by the flash file system, which eliminates the need for a backup power supply unit, and thus produces the benefit of simplifying the power supply circuit configuration of the flash file system.

[0023]
In the following, we describe Embodiment 5 of the present invention. Fig. 25 illustrates a hardware configuration that implements the present invention. Fig. 26 shows the memory organization inside the flash memory of the present invention. Fig. 27 is a flowchart for the writing of data. For convenience, we begin by explaining Fig. 26. In the figure, Reference Item 311 denotes a memory area in units

of 512 bytes (4 kilobits). Because it is a sector into which actual data is written, the memory area is referred to as a physical sector. By contrast, it is assumed that the supplier of data to be stored keeps track of data by means of virtual sectors, which are called logical sectors. All physical sectors 311 and logical sectors are assigned numbers, which have no correlation among them. Reference Item 312 denotes an erasure block that serves as a unit of batch erasure of data. The size of this block varies from one type of memory to another. For example, if an erasure block unit is 16 kilobytes, 32 physical sectors make up one block. Reference Item 313 denotes a memory chip. In the case of a 4-megabit chip with a 16-kilobyte erasure unit, the chip is divided into 32 blocks and 1024 sectors. If memory can be erased only in units of chips, one chip equals one block, and if memory can be erased in units of physical sectors, the number of blocks is equal to the number of physical sectors. In flash memory, random access is possible in a given block, if the block has undergone an erasure operation. However, if a block is written, it cannot be rewritten unless it is subjected to erasure. Therefore, in a memory in which multiple sectors make up a block, sectors should be erased only if all of the sectors have been written. In the following explanation, it is assumed that the flash memory is a 4-megabit chip, and that an erasure block is 16 kilobytes (128 kilobits). Fig. 26 illustrates memory with this organization. Fig. 25 illustrates a hardware configuration using this memory chip 313. In the figure, Reference Item 301 denotes flash memory, which is a write-data memory area; 302, an access controller that accesses the flash memory 301; 303, a processor that manipulates stored data and status data; 304, a logical sector table in which physical sector numbers are recorded to provide reference to which physical sector in the flash memory 301 the data for a stored logical sector number is stored; 305, a physical sector table that provides reference to logical sector numbers that are stored in the logical sector 311 of the flash

memory 301; 306, a block table that stores status information on the various blocks, where information such as whether a given block is available, and how many sectors have been written into that block, is recorded; 307, a write buffer that temporarily holds write data because in flash memory, the speed of writing is significantly slower than the speed of reading, so that the bus mastership can be returned speedily to the data supplier side; and 308, a cleanup data buffer that temporarily holds cleanup data in the cleanup routine, which will be discussed later. Fig. 27 is a flowchart that describes the operation of the processor 303 of Fig. 25, for the performance of writing operations in the configuration of Fig. 25. In the following, we explain the operation with reference to this flowchart. The data to be written is stored in units of 1 sector = 512 bytes. Thus, any data less than 512 bytes is also stored in a 512-byte memory area. Numbers, called logical sector numbers, are assigned to sector-by-sector data. The data supplier side only needs to keep track of the logical sector numbers. Suppose now that the data supplier side makes a request for writing one sector of data. A logical sector number is assigned to the data. Subsequently, even when rewrite operations occur, the data is identified in terms of the logical sector number. On the other hand, the data storage side determines where this data is to be stored. The location where the data is to be stored is the first unwritten to sector in the block that is indicated by the write pointer. In other words, it is the next sector from a previously written physical sector. For example, in Fig. 26, if the result of one previous write operation is written into the third physical sector of the first block, the next write operation will be performed on the fourth physical sector of the first block. The processor 303 determines the location of data storage by accessing the block table. This operation is illustrated in Step (a) in the flowchart. In Step (b), if the write operation poses no problem, the write sector count for the block is incremented by 1, and then the data

is written. The actual write access control is performed by the access controller 302. After the write operation is completed, the sector numbers that are written to the physical sector table and logical sector table are recorded, respectively. If the operation involves a rewriting of a previously written logical sector, the logical sector number that was previously written to the physical sector table is erased. The purpose of this operation is to indicate that the data in the affected physical sector is no longer valid. The cleanup routine in Step (c) is an operation routine that is invoked when the all the sectors in the block indicated by the write pointer are already written to and are unavailable for further write operations. This will be explained with reference to the flowchart to be described later. Step (d) is an operation wherein the block is broken down or control shifted to the cleanup routine, and write operations are disabled, and consequently, any writing to that block is canceled and a write operation is performed on the next block. On the other hand, during read access, the logical sector table is referenced using a logical sector number of the sector in which the requisite data is stored, a physical sector number is determined, and the required data is read. In the following, we explain the cleanup routine. The reason that a cleanup routine is needed is that the write methods that have been described thus far involve the writing to another physical sector when data in a given file is to be rewritten. In other words, although the data that was written before a rewrite operation is performed is no longer needed, but it remains stored in the memory, and it must be erased. However, given that flash memory has a finite erasure count, erasing data every time it becomes unneeded results in a decrease in its useful life. For this reason, erasure is performed by means of the cleanup routine, which is executed when a block has become full with write data. Specifically, clean-up is performed according to the flowchart shown in Fig. 28. In the following, we describe the various steps in the flowchart:

(a) Save all data once in the block to be cleaned up to the cleanup data buffer 308 of Fig. 25. (b) Erase the block when the block becomes erasable because of the saving. (c) By referencing the physical sector table for the sectors contained in the clean-up data buffer, determine whether the data is still needed or no longer needed. (d) If the data is still needed, write it again into the block. This concludes a description of the operation of an embodiment of the present invention. According to this Embodiment, the erasing of the various erasure blocks in the flash memory does not concentrate locally, but instead it is performed sequentially, the benefit of a prolonged life can be expected. In addition, this Embodiment produces the benefit of compensating for the drawback of a slow write speed in the flash memory. In addition, because unneeded data is efficiently erased, as a semiconductor disk, the Embodiment provides the benefit of constantly maintaining the storage capacity of the disk.

[0024]
In the following, we describe another embodiment. Fig. 29 shows an embodiment of a system that determines the deterioration of flash memory. In the figure, Reference Item 401 denotes a flash memory cell; 402, an erasure-time measurement timer that measures the length of time from the beginning to the end of erasure control; 403, an erasure control circuit that performs controls so that the data written in the flash memory cell 401 can be erased; 404, a deterioration level-tracking table that stores the level of deterioration of the flash memory cell 401; 405, a controller that coordinates and controls these items; 406, an activation signal that simultaneously starts the erasure-time measurement timer 402 when the deterioration-tracking controller 405 causes the erasure control circuit 403 to commence the erasure process; 407, a termination signal that enables the erasure control circuit 403 to inform the erasure-time measurement timer 402 of the termination of the erasure process; and 408, measurement

data from the erasure-time measurement timer. Fig. 30 is a flowchart that describes the operation of the controller 405 of Fig. 29. The determination of flash memory deterioration is performed according to this flowchart. A description of the operation of this process follows with references to Figs. 29 and 30. First, when the need for an erasure operation on a certain memory cell 401 arises for overall system controls, the controller 405 receives an erasure request (Fig. 30 a). Then, the controller 405 instructs the erasure control circuit 403 in terms where the erasure is to be performed, and instructs the circuit to begin the erasure process (Fig. 30 b). Upon reception of these instructions, the erasure control circuit 403 begins an erasure operation on the applicable memory cell 401 (Fig. 30 c). Simultaneously, the controller 405 starts the erasure-time measurement timer 402 (Fig. 30 d). The controller 405 waits until the erasure process terminates. When completing the erasure of the memory cell 401, the memory control circuit 403 notifies the controller 405 of the event (Fig. 30 e). By referencing the erasure-time measurement timer 402, the controller 405 recognizes the amount of time spent on the erasure process, determines the extent of deterioration, and assigns a corresponding number to the memory area. For example, the extent of deterioration can be divided into eight levels, so that if the amount of time consumed is the same as an unused status, a "0" value is assigned, and if deterioration has progressed extensively to render the memory area unusable, a value "7" is assigned, and intervening levels are assigned numbers "1" through "6". The number assigned to the particular memory area is stored in the deterioration level tracking table 404 (Fig. 30 f). The division of the extent of deterioration into eight levels permits the assignment of one byte to a memory area, which facilitates tracking. The various memory areas are handled based on the deterioration level tracking table. Each time the extent of deterioration of an area rises by one level, data from a lower deterioration level area is transferred to

this area in order to arrest the progression of deterioration, so that the extent of deterioration for all memory areas will average out. Further, any area that has reached the final level of deterioration is disabled from use. This is the type of processing that is performed by the controller that provides coordination and control for the entire memory system. Fig. 31 shows a flowchart depicting the operation of the memory system controller. We now explain the flowchart in sequence with reference to that figure. A write access request is made from the host system to the memory system. When performing a data write operation, if the controller for the memory system determines that an erasure operation is needed (Fig. 31 a), the controller sends erasure and erasure-tracking requests to the controller 405 of Fig. 29 (Fig. 31 b). Upon termination of the erasure operation, the controller references the erasure level-tracking table of Fig. 29 for the extent of deterioration of the memory area on which the erasure operation was performed (Fig. 31 b). If the deterioration is advanced, the controller searches for an area which is least deteriorated and has not undergone any data swapping (Fig. 31 c), and transfers data to effect swapping (Fig. 31 d). The effective method for making the determination that a memory area is least deteriorated and its data has not been swapped would be the use of the swapping flag that was described in conjunction with Embodiment 1. The reason is that because the data contained in an area that has received a swapping operation is not data that does not deteriorate (i.e., not data that undergoes little rewriting), such data should not be used in swapping for deterioration-averaging purposes, and this fact is explicitly indicated by raising a swapping flag. It should be noted that, although this Embodiment divides controls into overall control of the memory system and control on the deterioration determination system, with a specific controller used for each type of control, the controller 405 can double as a controller for the memory system. Further, although the erasure time determination

timer 402 is configured as a piece of hardware, alternatively, erasure time can be measured by the controller on a software-controlled basis. These modifications should be made if the number of components in the memory system is to be reduced. According to this Embodiment, the table for deterioration-tracking can be changed to the deterioration-tracking table that was used in the preceding embodiments, which can significantly reduce the size of the memory area to be used. Further, because erasure time provides a more direct measure of deterioration than an erasure count from the standpoint of determining the extent of deterioration of memory, the use of a deterioration-tracking table can produce the benefits of an accurate deterioration assessment and the averaging out of deterioration.

[0025]
In the following, we describe another embodiment of erasure count tracking using the embodiment that was illustrated using Fig. 24. Whereas the erasure count-tracking method used in the previous embodiment involves a detailed tracking of erasure counts by incrementing the count each time an erasure is performed, in the present embodiment, lower bits are truncated in order to save the amount of memory required in the erasure count-tracking table. In the embodiment shown in Fig. 24, in the normal usage condition, erasure counts, including the least significant bit, are stored in volatile memory, such as SRAM and DRAM, and this information is transferred to the flash memory when the power is shut off. In the embodiment of Fig. 32, however, one or more bits, beginning with the least significant bit, are truncated during the transfer process (Fig. 32 a) or it is transferred after carrying (Fig. 32 b). The lower bits that are truncated should be limited to an extent that does not impair the accuracy of the recognition of deterioration based on an erasure count. An optimal value should be determined based on the limit on flash memory erasure counts, and the maximum value of the erasure count that is

rounded off due to the truncation of bits should not exceed 1% of the guaranteed value of the flash memory erasure count. In other words, if a guaranteed erasure count is 10,000 times, 6 bits (maximum value: 63), which is less than 100, should be the limit of truncation. Although this embodiment reduces the accuracy of erasure counts, because the error is held to less than 1% of the guaranteed value for flash memory erasure counts, and given that originally the accuracy of the guaranteed value is subject to a one-digit error, for the purposes of determining whether or not stored data has a strong likelihood of being erased, this does not result in an essential problem. This approach can yield significant benefits in reducing the size of flash memory used for erasure count-tracking purposes.

[0026]
In the following, we describe another embodiment that reduces the size of the table area. Fig. 33 is a configuration diagram of a memory system from which physical sector and logical sector tables, explained in the preceding embodiments, are omitted. Instead of those tables, an address translation table 411 is provided. The address translation table 411 does not translate any address input for memory areas that have not undergone data swapping for deterioration-averaging purposes. When the address of a memory area that has undergone data swapping is input, the address translation table is used to output the address of the destination of the swapping. Therefore, this embodiment does not require physical sector or logical sector tables. Basically, when a request for access to an address is made by the system, this embodiment directly associates the physical address in the memory for access, and if the deterioration of that area advances and data swapping is made for deterioration averaging, the accessed address is subject to address translation. This Embodiment, as Embodiment 1, can be applied only to situations where file data storage units (sectors) are the same as erasure units. The size of the address translation

table depends on the size of the area that can be rendered translatable. In other words, reducing the size of the address translation table reduces the size of the memory that can be translated, and allocating a large area permits the translation of a large number of area addresses. This factor directly relates to the life of the system. Fig. 33 shows a configuration of a modified version of Embodiment 1, and it can also be applied to embodiments in which the type of deterioration level-tracking table described in Fig. 29 is provided instead of the erasure-tracking table 7. This Embodiment provides the benefits of simplified data tracking and a significant reduction in the table area.

[0027]

In the following, we describe an embodiment in which an indicator that indicates that the system is running is provided. Fig. 34 is an internal configuration diagram showing this Embodiment. Based on Fig. 17, the numbers in this figure are identical to the previously described numbers in Fig. 17. Other numbers in the figure include the following: Reference Item 421 denotes an output port signal for the processor 3, which indicates that data is being transferred from the write buffer 9 to the flash memory 1; and 422, an indicator that indicates by means of light emission that data transfer is in progress, based on the indicator 421. An appropriate indicator that can be used is a light-emitting diode (LED). Fig. 35 is an external view of this Embodiment, in which (1) represents a card-shaped overall appearance, and (2) shows an example of the Embodiment in use. In the figure, Reference Item 423 denotes an IC card, which is an auxiliary memory unit of the present invention; 424, a connector; 425, LED; and 426, a host personal computer. In Fig. 34, when the host personal computer sends a write access request to the auxiliary memory unit 205 through the standard I/O bus 201, the processor 3 processes so that write data is stored in the write buffer 9. Upon completion of this task, the

processor outputs a signal indicating the completion of the write process to the standard bus 201. Subsequently, the processor 3 transfers and stores the data stored in the write buffer 9 to the flash memory 1. During this processing, the processor 3 activates the output port signal 421, and causes the indicator 422 to emit light. Upon completion of the transfer process from the write buffer 9 to the flash memory 1, the processor 3 makes the output port 421 inactive, and stops the light emission from the indicator 425. Fig. 35 (1) is an example where the Embodiment is applied in an IC card form, wherein the indicator 425 is attached to the side opposite the connector, and the lighting of the indicator 425 can be checked under the condition in which the IC card is attached to the personal computer. Fig. 35 (2) illustrates the manner in which the IC card is actually inserted into a laptop computer, so that the user can perform operations by checking to see whether the indicator 425 is lit or off. The user needs to check the lighting of the indicator basically only when the power supply is shut off. This Embodiment permits a relatively simple circuit organization, and provides the benefits of ease of viewing the indicator and preventing human error on the part of the user.

[0028]

[Effects of the Invention]

According to the present invention, in the data management method for an auxiliary memory unit using flash memory, even when a specific logical sector address is subject to frequent rewrite operations, physically the same memory area is not used, and as the number of erasures increases, data is swapped with data in a low erasure count area so as to average out any increase in the erasure count, and this provides the benefit of an increased system life. In addition, it suffices that the number of data memory elements used match the actual storage capacity, and this invention does not require redundant memory elements.

Further, in addition to data areas, information retention areas and data buffer areas are provided in the memory, thereby reducing the number of elements in the peripheral circuitry and this fact significantly contributes to the downsizing of the system as a whole. The information device that incorporates this flash file system can be configured so that while it originally runs on an HDD as an auxiliary memory unit, in actuality it runs on a storage unit using flash memory. In this manner, the invention permits the use of a flash file system as an auxiliary memory unit without the need for modifying the hardware of a commonly available information device. Due to the fact that there is a limit on the number of times flash memory can be rewritten, the life of a memory system significantly affects its reliability; however, the invention provides functions of recognizing the usage limit being reached and notifying the user, which can improve the reliability of the memory system. In addition, because flash memory has a slow write speed, if the power for the host system is shut off but the flash file system still has processing to do, a backup battery is used to continue with the processing or the power supply unit for the host is appropriately controlled in order to prevent data loss. Further, the extent of deterioration of the flash memory is determined according to erasure time, and where deterioration is extensive, data stored in a less deteriorated area can be stored to that area, thus retarding the progression of deterioration. The invention permits the recognition of flash memory deterioration in terms of how long or how short is erasure time, which permits an accurate evaluation of the extent of deterioration, and if the extent of deterioration is stored in terms of levels, information on the extent of deterioration can be stored in a relatively small memory size, which is beneficial. Further, by omitting physical sector and logical sector tables and by providing a sector number translation table, the size of the table area can be reduced depending upon the type of the system used, and this yields the benefit of reducing

the required storage area for informational data other than file data. In addition, in view of the fact that only a prompt alone that appears on the display screen of the host personal computer does not clearly indicate the operating status of the auxiliary memory unit, in the case of a system that does not include power supply control of the type described above, the provision of an indicator on the auxiliary memory unit can prevent the inadvertent shutting off of the power when the auxiliary memory unit is still running.

4. BRIEF DESCRIPTION OF DRAWINGS

[Fig. 1] A hardware configuration diagram of Embodiment 1 of the present invention.
[Fig. 2] A memory configuration diagram of the flash memory chip in Embodiment 1 of the present invention.
[Fig. 3] A main routine flowchart that describes the operation of Embodiment 1 of the present invention.
[Fig. 4] An erasure-tracking routine flowchart that describes the erasure-tracking operation of Embodiment 1 of the present invention.
[Fig. 5] A hardware configuration diagram of Embodiment 2 of the present invention.
[Fig. 6] A memory configuration diagram of the flash memory chip in Embodiment 2 of the present invention.
[Fig. 7] A main routine flowchart that describes the operation of Embodiment 2 of the present invention.
[Fig. 8] A clean-up routine flowchart that describes the sector clean-up operation of Embodiment 2 of the present invention.
[Fig. 9] An erasure-tracking routine flowchart that describes the erasure-tracking operation of Embodiment 2 of the present invention.
[Fig. 10] A hardware configuration diagram of Embodiment 3 of the present invention.

[Fig. 11] A memory configuration diagram of the flash memory chip in Embodiment 3 of the present invention.
[Fig. 12] An erasure-tracking routine flowchart that describes the erasure-tracking operation of Embodiment 3 of the present invention.
[Fig. 13] A hardware configuration diagram of Embodiment 4 of the present invention.
[Fig. 14] A configuration diagram of the flash memory chip with a table provided for each erasure block.
[Fig. 15] A configuration diagram of the flash memory chip with an information storage area provided, separate from a data area.
[Fig. 16] A configuration diagram of the flash memory chip with a buffer area provided, separate from a data area.
[Fig. 17] A hardware configuration diagram of the interface part for the flash file system of Embodiment 4 of the present invention.
[Fig. 18] A configuration diagram of the information device of Embodiment 4 of the present invention, in which a flash file system is used as an auxiliary memory unit.
[Fig. 19] A configuration diagram of an embodiment of the present invention in which a usage limit is reported.
[Fig. 20] A flowchart of an embodiment of the present invention in which a usage limit is reported.
[Fig. 21] A configuration diagram of an embodiment of the present invention in which a backup battery is provided.
[Fig. 22] A configuration diagram of an embodiment of the present invention in which a power supply control signal is provided.
[Fig. 23] An embodiment of data storage in the flash memory with an information storage area provided, separate from a data area.
[Fig. 24] A configuration diagram of an embodiment from which the EEPROM of Fig. 13 is omitted.
[Fig. 25] A hardware configuration diagram of Embodiment 5 of the present invention.
[Fig. 26] A memory configuration diagram of the flash memory chip in Embodiment 5 of the present invention.

[Fig. 27] A main routine flowchart that describes the operation of Embodiment 5 of the present invention.
[Fig. 28] A clean-up routine flowchart that describes the sector clean-up operation of Embodiment 5 of the present invention.
[Fig. 29] A configuration diagram of an embodiment that seeks to prolong the life of the system by managing deterioration based on time.
[Fig. 30] An operating flowchart describing the deterioration management controller in the embodiment shown in Fig. 29.
[Fig. 31] An operating flowchart describing the memory system controller in the embodiment shown in Fig. 29.
[Fig. 32] An explanatory diagram of an embodiment that minimizes the amount of memory required by erasure-tracking tables.
[Fig. 33] A configuration diagram of an embodiment that adopts an address translation table that minimizes the amount of memory required by sector management tables.
[Fig. 34] A configuration diagram of an embodiment that provides an indicator that shows that the auxiliary memory unit is running.
[Fig. 35] An external view of an embodiment that provides an indicator that shows that the auxiliary memory unit is running.

[Numerics in Figures]
1 … flash memory
3 … processor
5 … logical sector table
6 … physical sector table
7 … erasure-tracking table
8 … status table
9 … write buffer
13 … erasure block
49 … write sector count table
51 … clean-up buffer
52 … physical sector
111 … single-chip microcomputer
112 … RAM core
113 … ROM core

114 ... EEPROM
115 ... DRAM or SRAM
116 ... table area
132 ... table storage area
142 ... buffer area
143 ... address counter
207 ... interface registers
227 ... BIOSROM
236 ... interface unit
241 ... error report register
253 ... backup battery
259 ... power supply busy signal
402 ... erasure-time measurement timer
404 ... deterioration level-tracking table
411 ... address translation table
421 ... output port signal
422 ... indicator
425 ... indicator (external view)

(Continued from the front page)

Inventors: Kenichi Kaki
292 Yoshida-cho, Totsuka-ku, Yokohama-shi, Kanagawa
Microelectronics Equipment Development Research Laboratory, Hitachi Ltd.
Takeshi Wada
5-20-1 Josuihon-cho, Kodaira-shi, Tokyo
Hitachi Ltd., Semiconductor Design Development Center
Takeshi Furuno
5-20-1 Josuihon-cho, Kodaira-shi, Tokyo
Hitachi Ltd., Semiconductor Design Development Center
Takashi Totsuka
5-20-1 Josuihon-cho, Kodaira-shi, Tokyo
Hitachi Ltd., Semiconductor Design Development Center

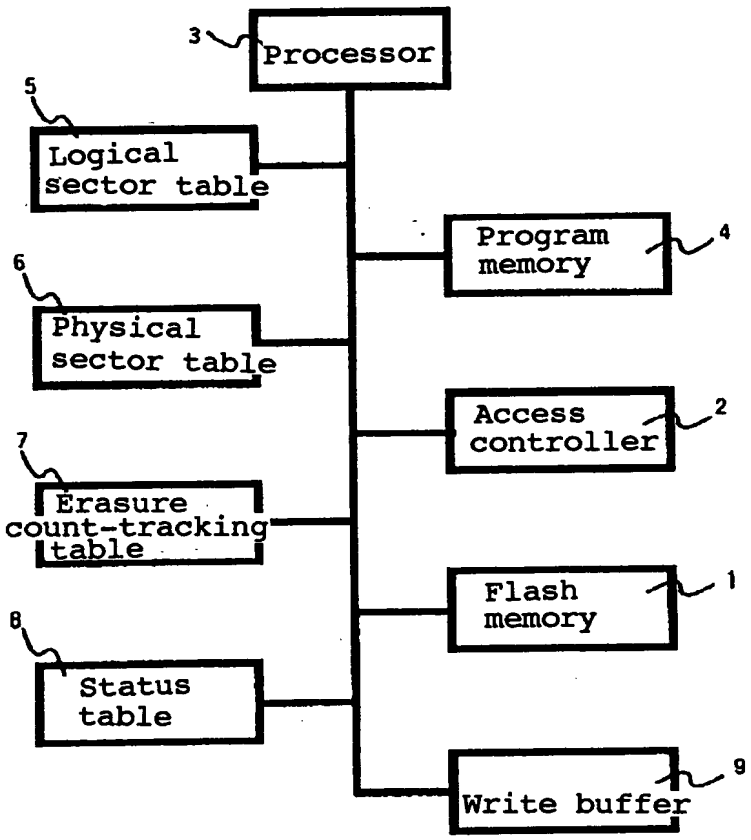

3
Processor
5
Logical sector table
Program memory
4
6
Physical sector table
Access controller
2
7
Erasure count-tracking table
Flash memory
1
8
Status table
Write buffer
9

Fig. 1

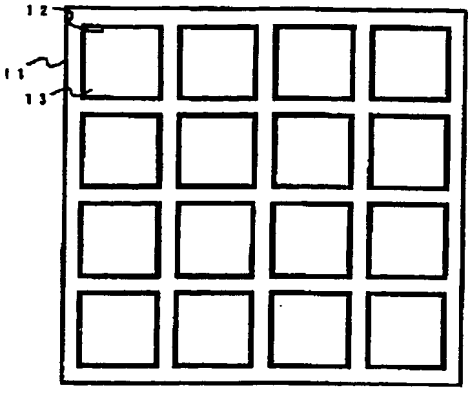


Fig. 2

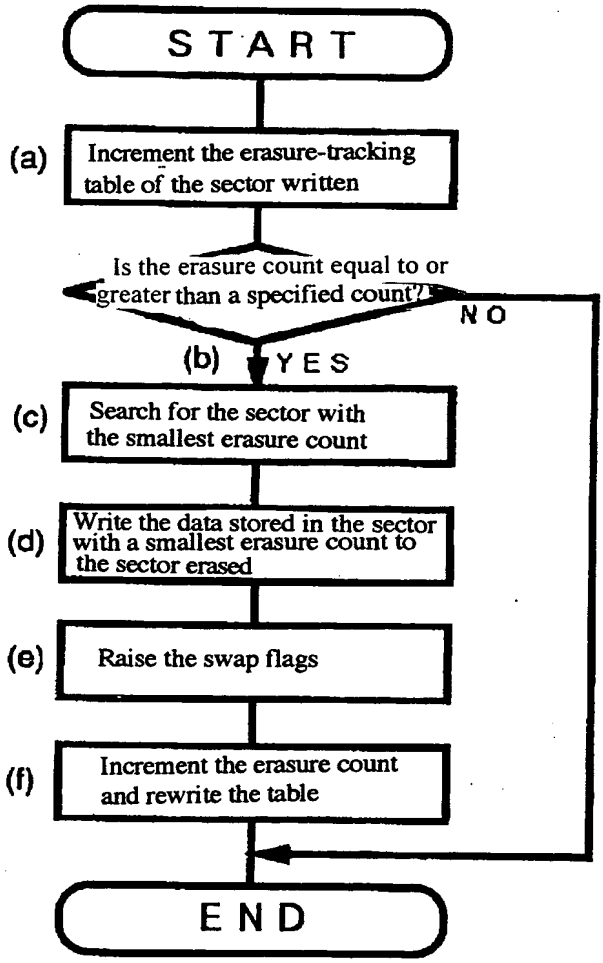


Fig. 4

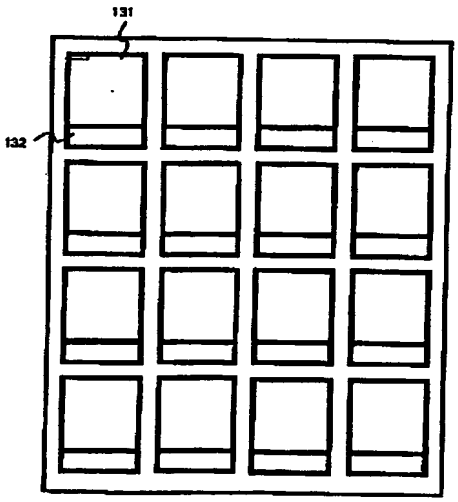


Fig. 14

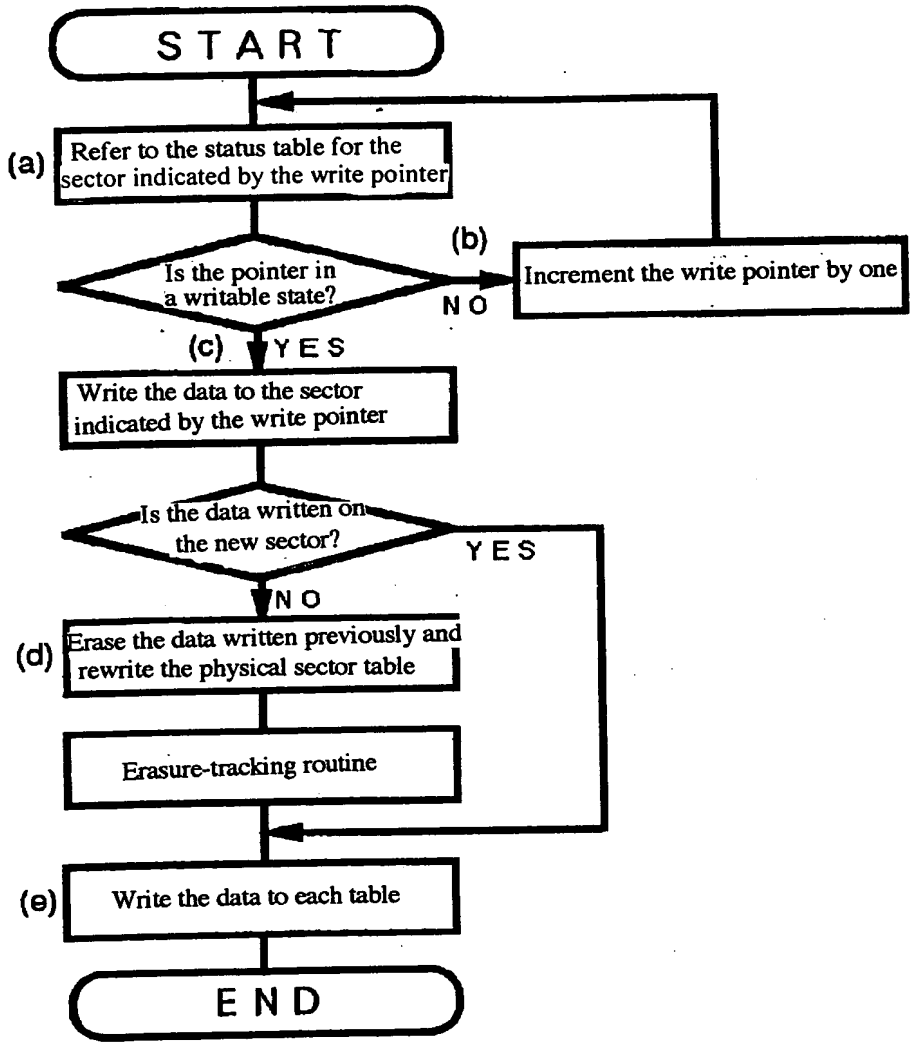

START
(a)
Refer to the status table for the sector indicated by the write pointer
(b)
Is the pointer in a writable state?
NO
Increment the write pointer by one
(c)
YES
Write the data to the sector indicated by the write pointer
Is the data written on the new sector?
YES
NO
(d)
Erase the data written previously and rewrite the physical sector table
Erasure-tracking routine
(e)
Write the data to each table
END

Fig. 3

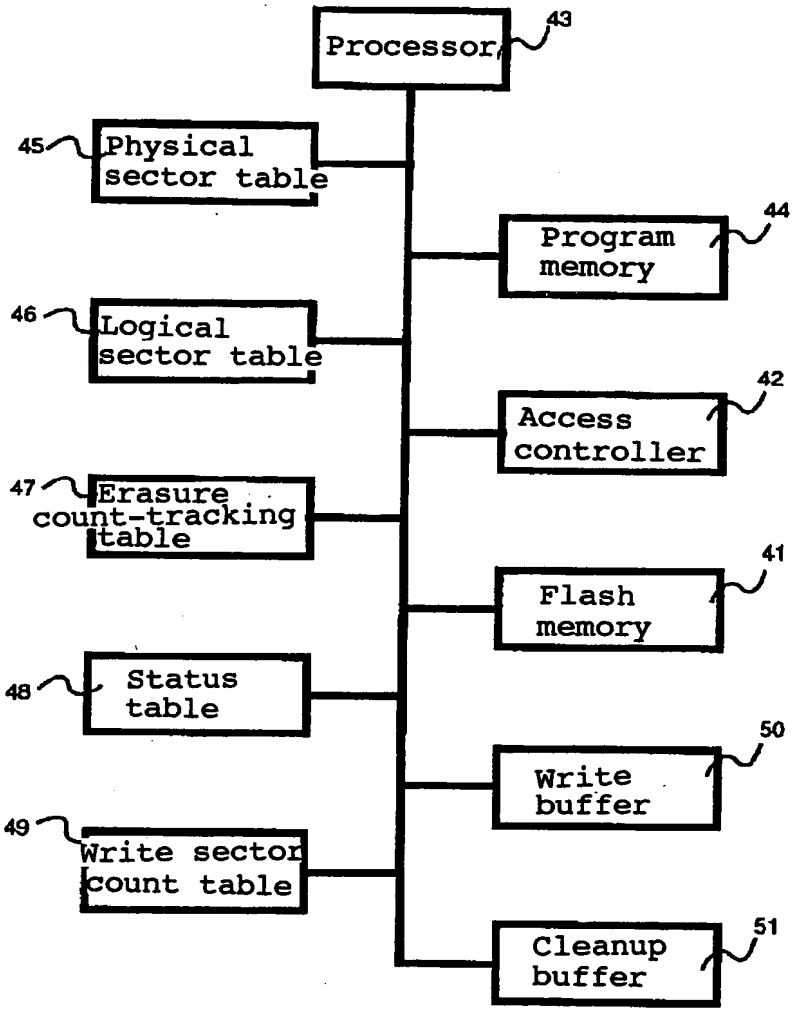

Processor
43
45
Physical sector table
Program memory
44
46
Logical sector table
Access controller
42
47
Erasure count-tracking table
Flash memory
41
48
Status table
Write buffer
50
49
Write sector count table
Cleanup buffer
51

Fig. 5

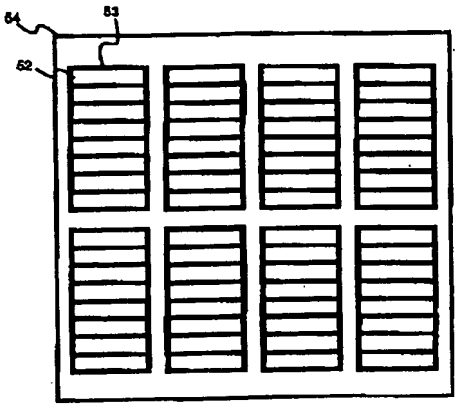


Fig. 6

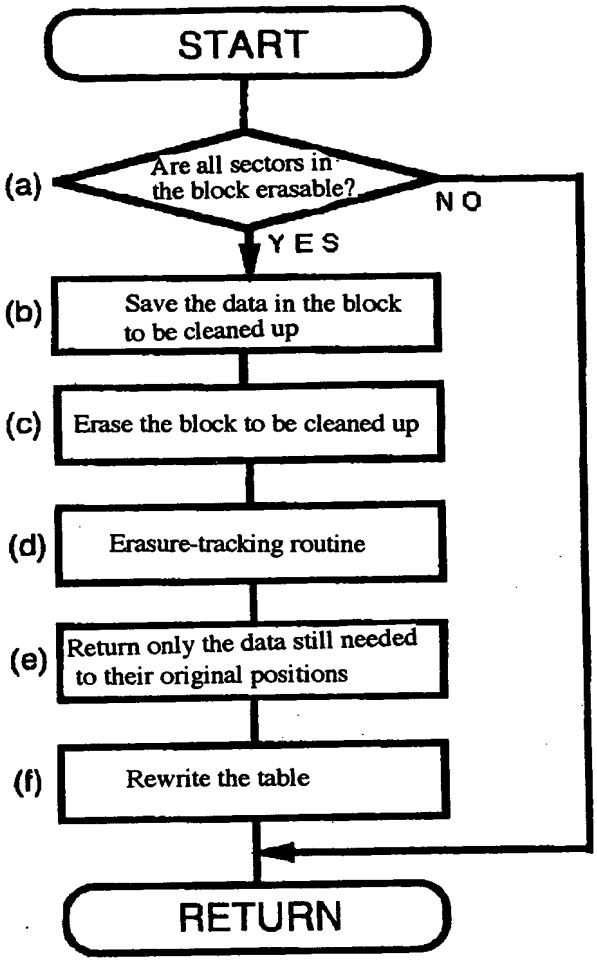
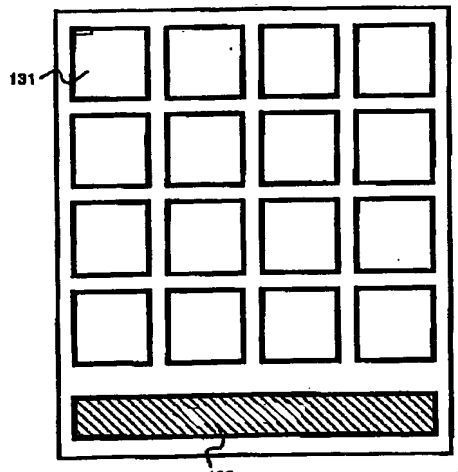


Fig. 8



138 Memory area for tables

Fig. 15

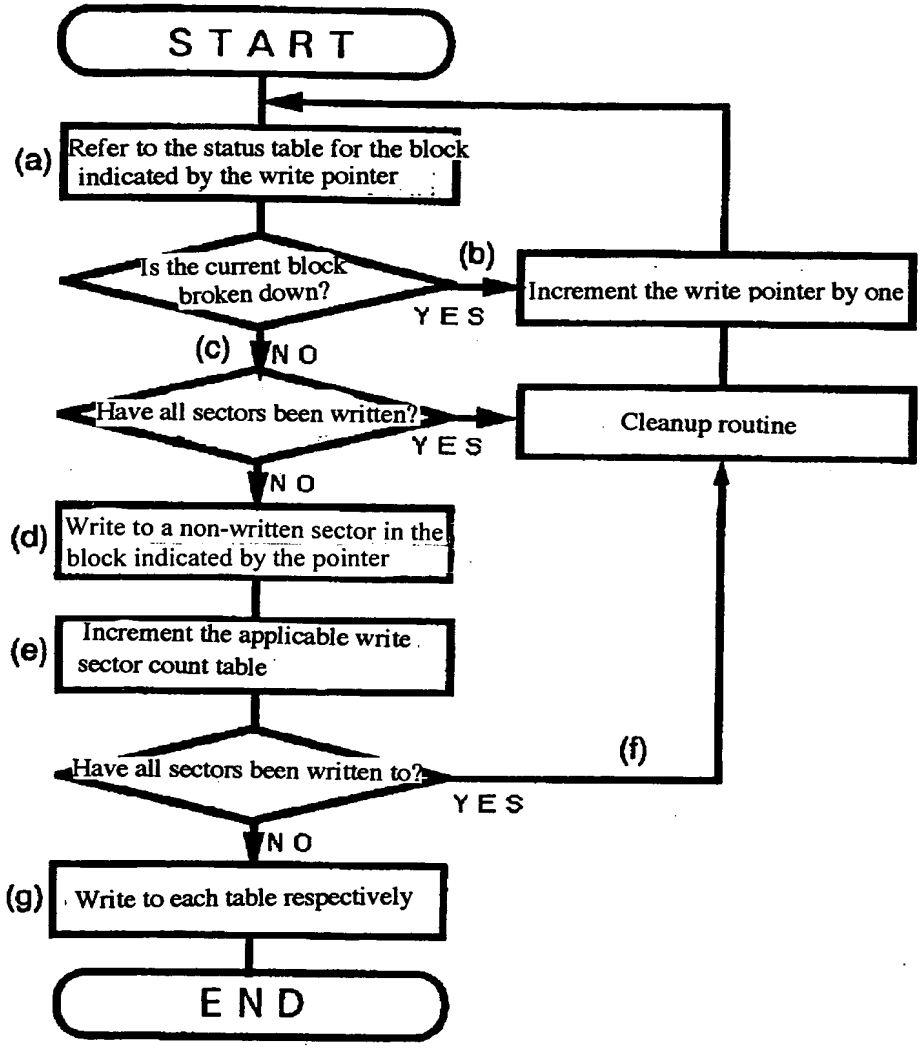

START
(a)
Refer to the status table for the block indicated by the write pointer
Is the current block broken down?
(b)
YES
Increment the write pointer by one
(c)
NO
Have all sectors been written?
YES
Cleanup routine
NO
(d)
Write to a non-written sector in the block indicated by the pointer
(e)
Increment the applicable write sector count table
Have all sectors been written to?
(f)
YES
NO
(g)
Write to each table respectively
END

Fig. 7

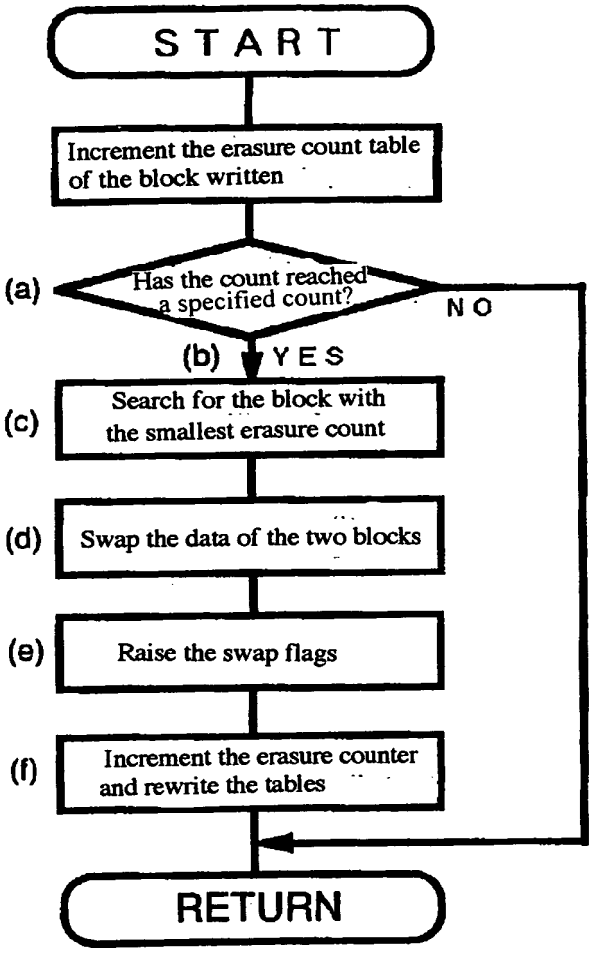


Fig. 9

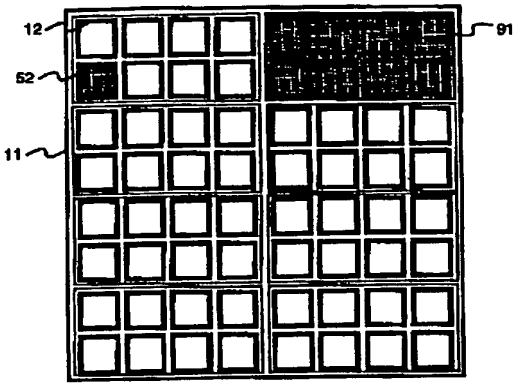


Fig. 10

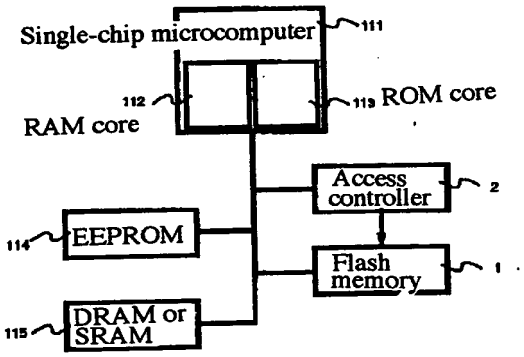


Fig. 13

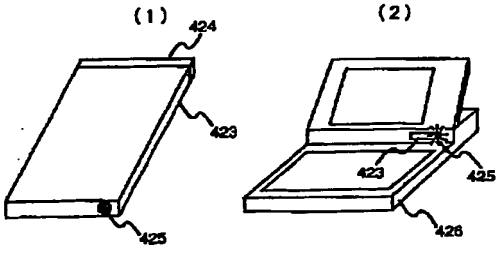


Fig. 35

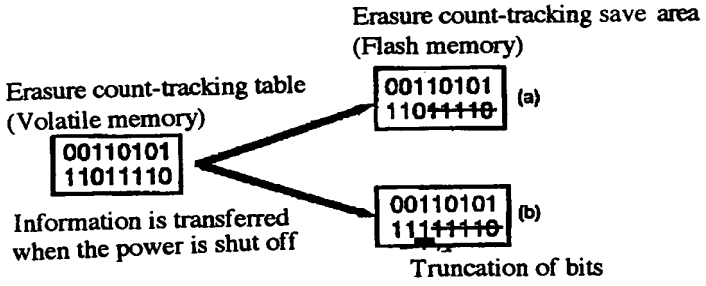


Fig. 32

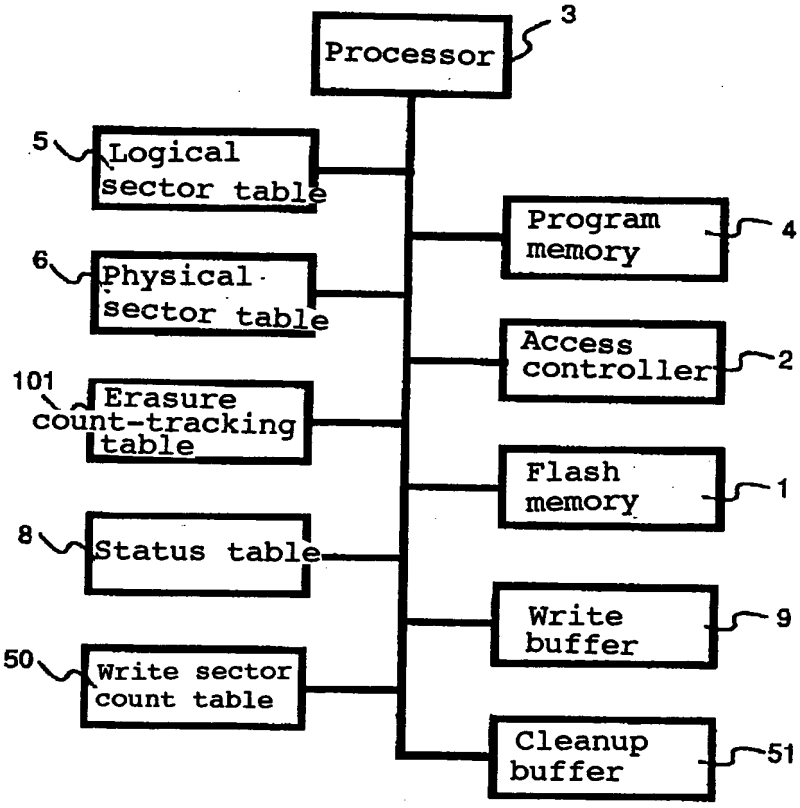

Processor
3
5
Logical sector table
Program memory
4
6
Physical sector table
Access controller
2
101
Erasure count-tracking table
Flash memory
1
8
Status table
Write buffer
9
50
Write sector count table
Cleanup buffer
51

Fig. 11

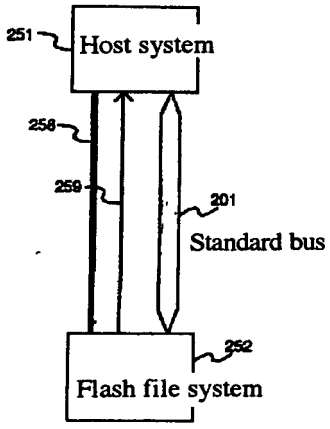

251
Host system
258
259
201
Standard bus
252
Flash file system

Fig. 22

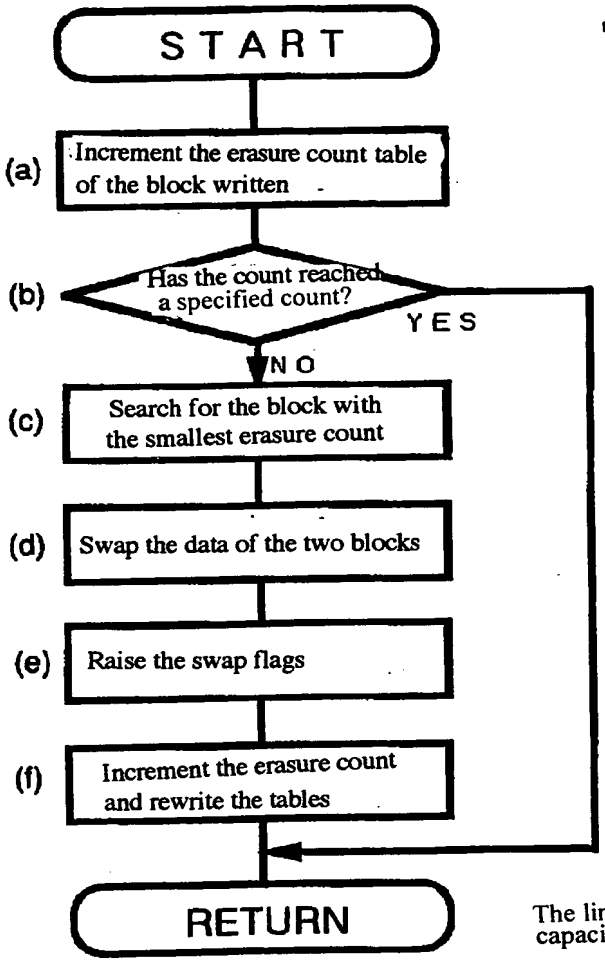


Fig. 12

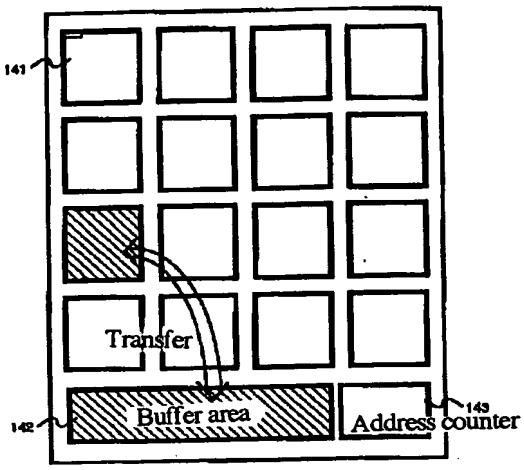


Fig. 16

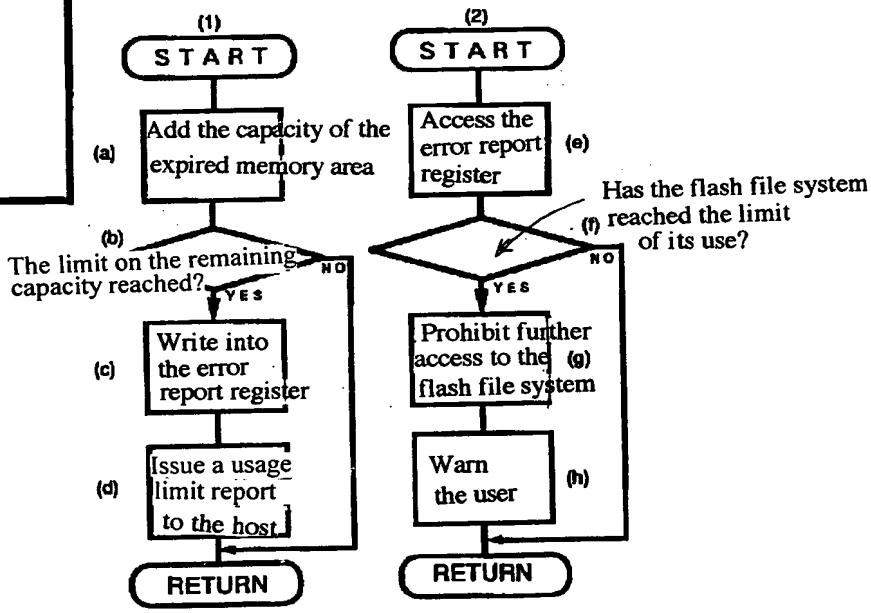


Fig. 20

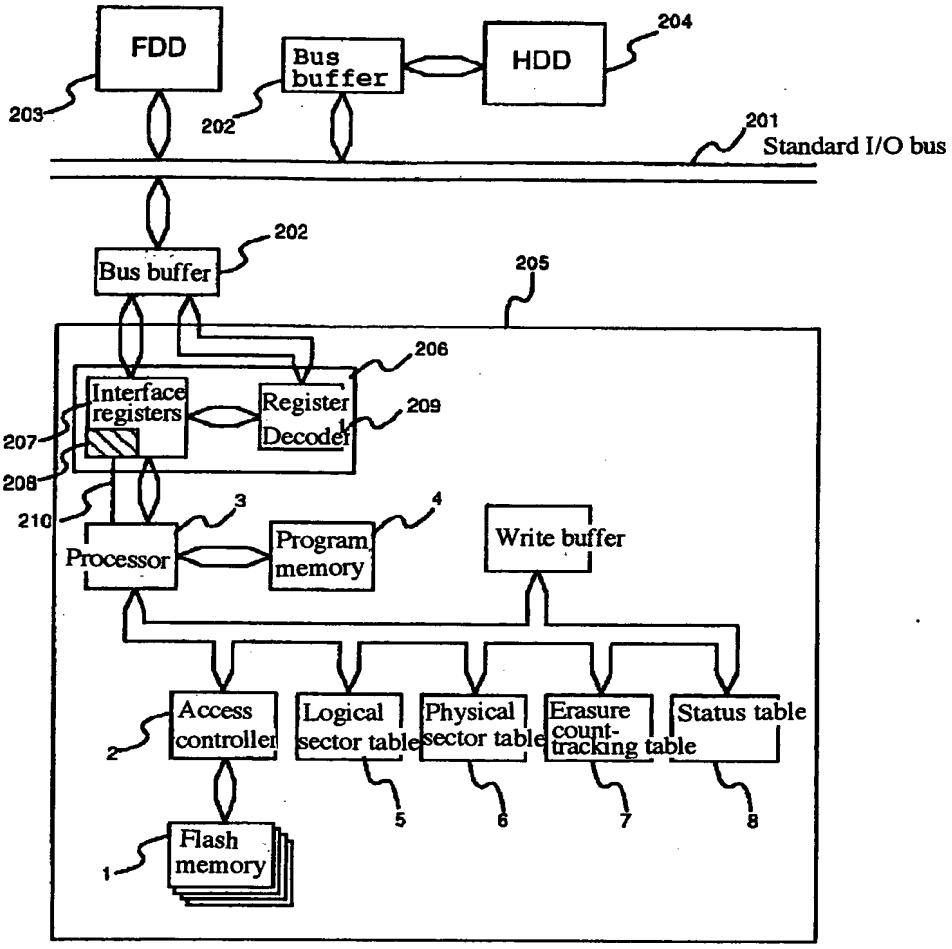

FDD
Bus buffer
HDD
204
203
202
201
Standard I/O bus
Bus buffer
202
205
206
Interface registers
Register Decoder
209
207
208
210
3
4
Processor
Program memory
Write buffer
Access controller
Logical sector table
Physical sector table
Erasure count-tracking table
Status table
2
5
6
7
8
Flash memory
1

Fig. 17

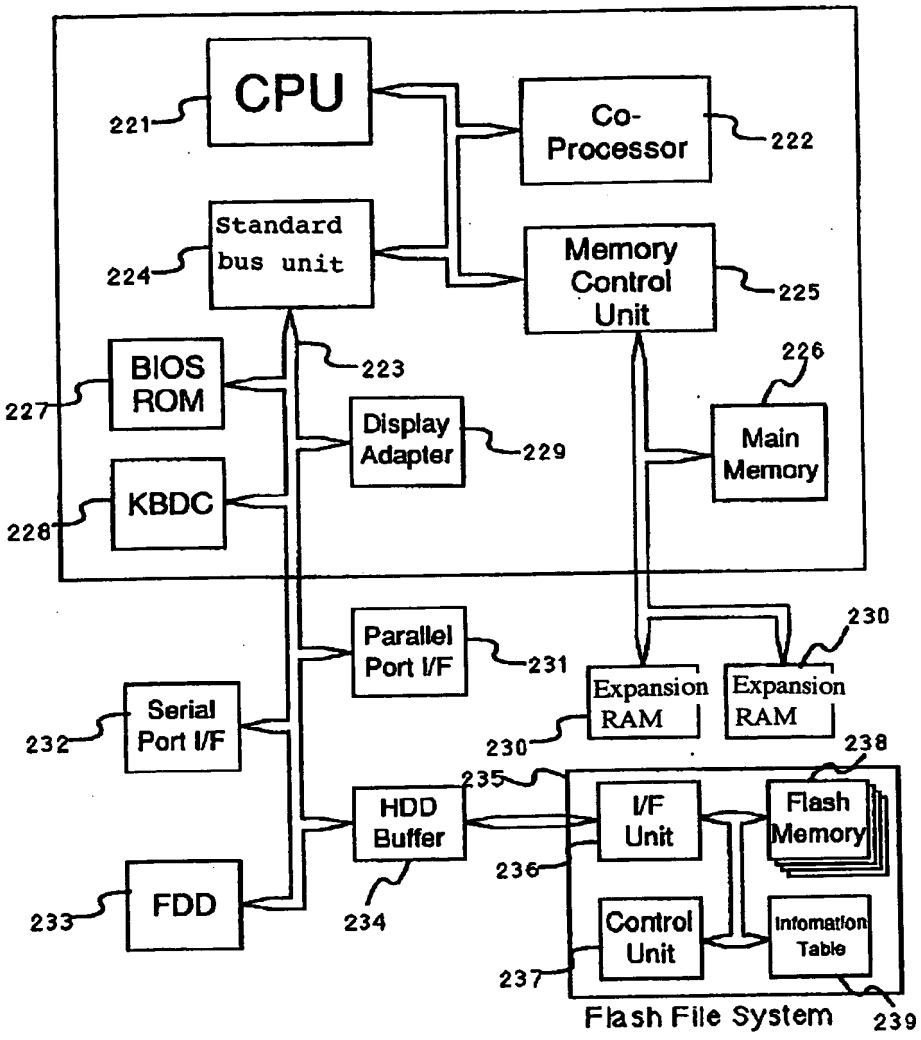

CPU
221
Co-
Processor
222
Standard
bus unit
224
Memory
Control
Unit
225
226
223
BIOS
ROM
227
Display
Adapter
229
Main
Memory
KBDC
228
Parallel
Port I/F
231
230
Expansion
RAM
Expansion
RAM
230
Serial
Port I/F
232
238
235
HDD
Buffer
I/F
Unit
Flash
Memory
236
FDD
233
234
Control
Unit
Infomation
Table
237
Flash File System
239

Fig. 18

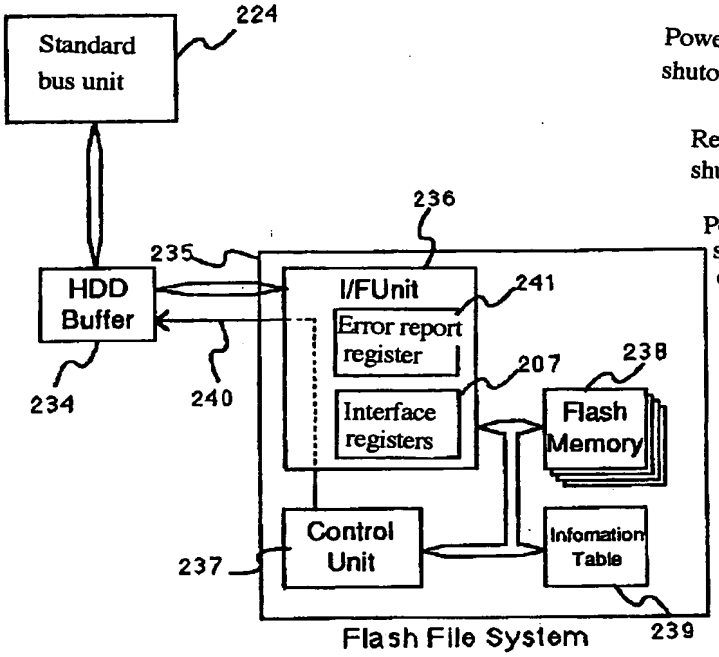


Fig. 19

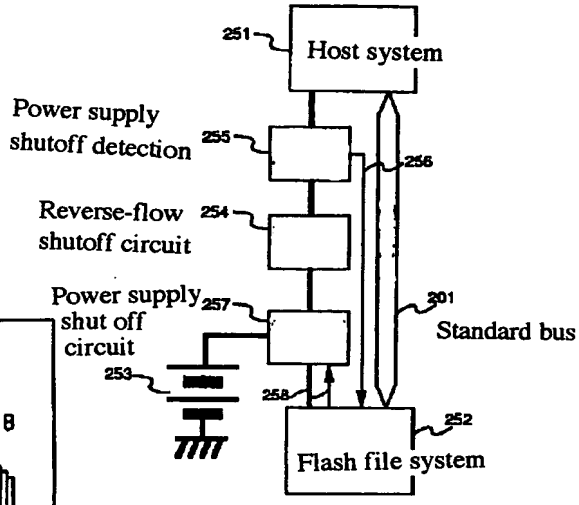


Fig. 21

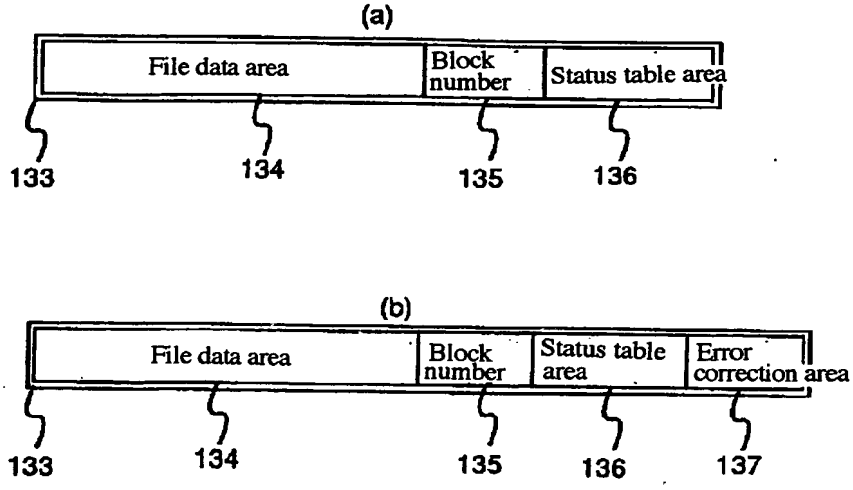


Fig. 23

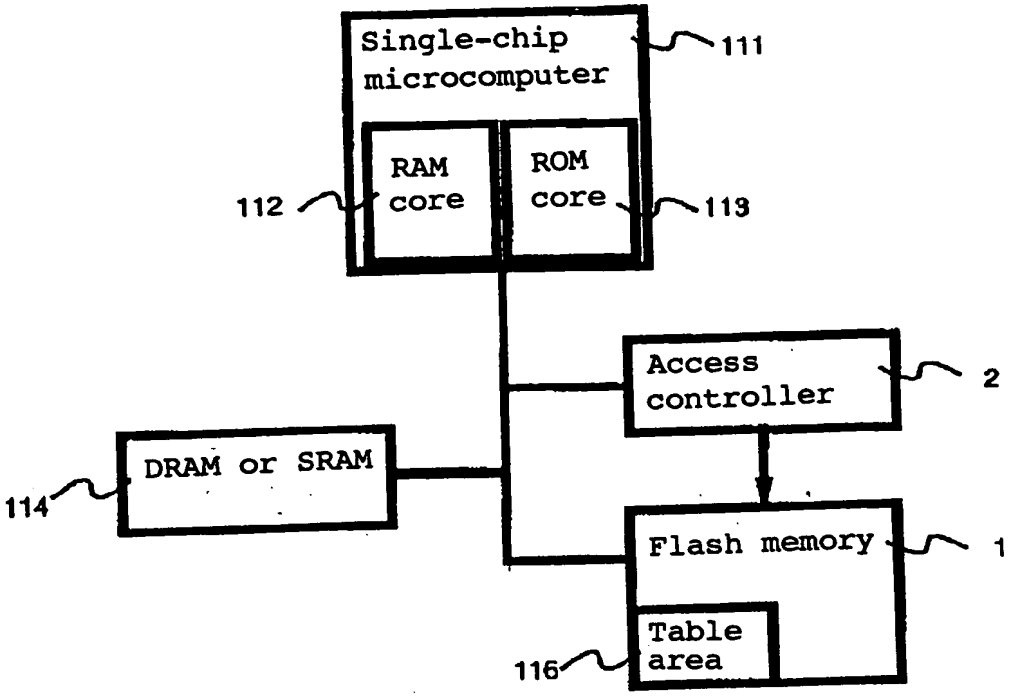


Fig. 24

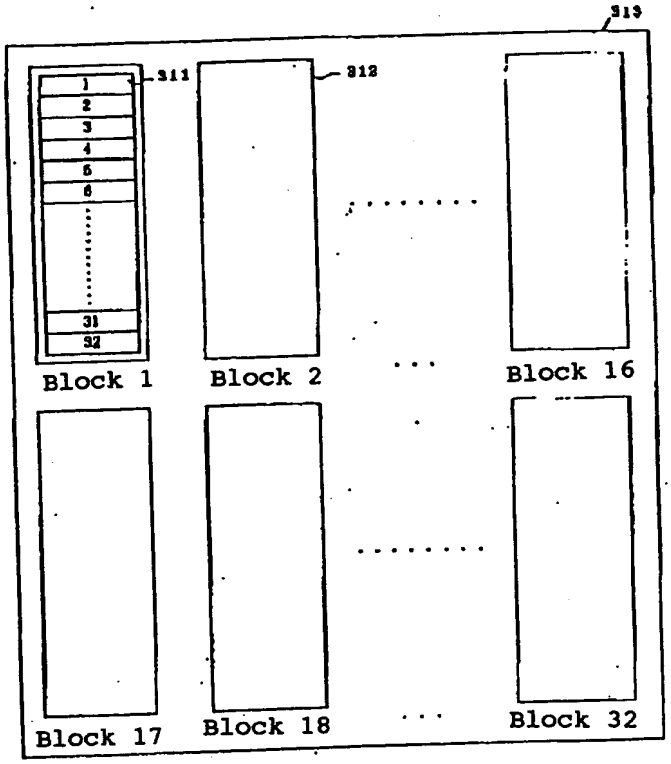


Fig. 26

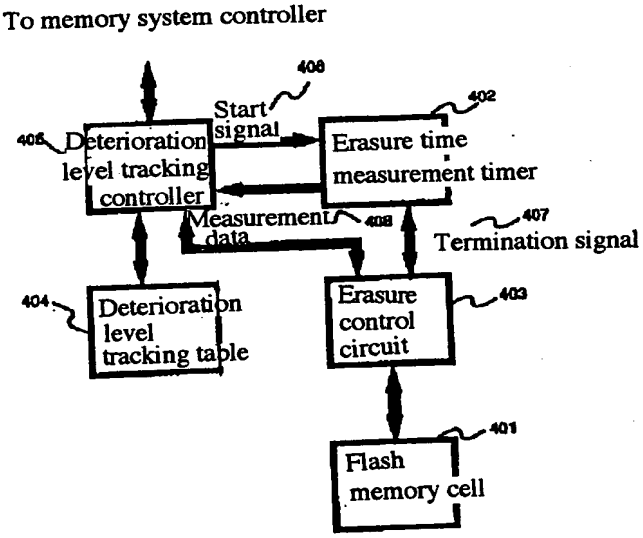


Fig. 29

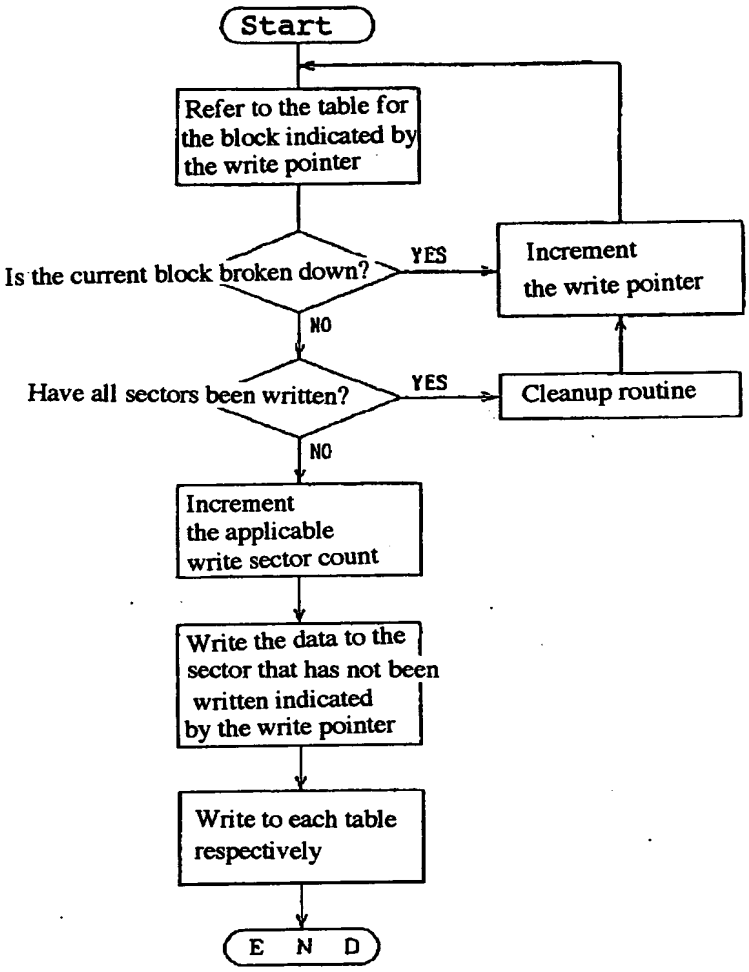

Start
Refer to the table for the block indicated by the write pointer
Is the current block broken down?
YES
Increment the write pointer
NO
Have all sectors been written?
YES
Cleanup routine
NO
Increment the applicable write sector count
Write the data to the sector that has not been written indicated by the write pointer
Write to each table respectively
E N D

Fig. 27

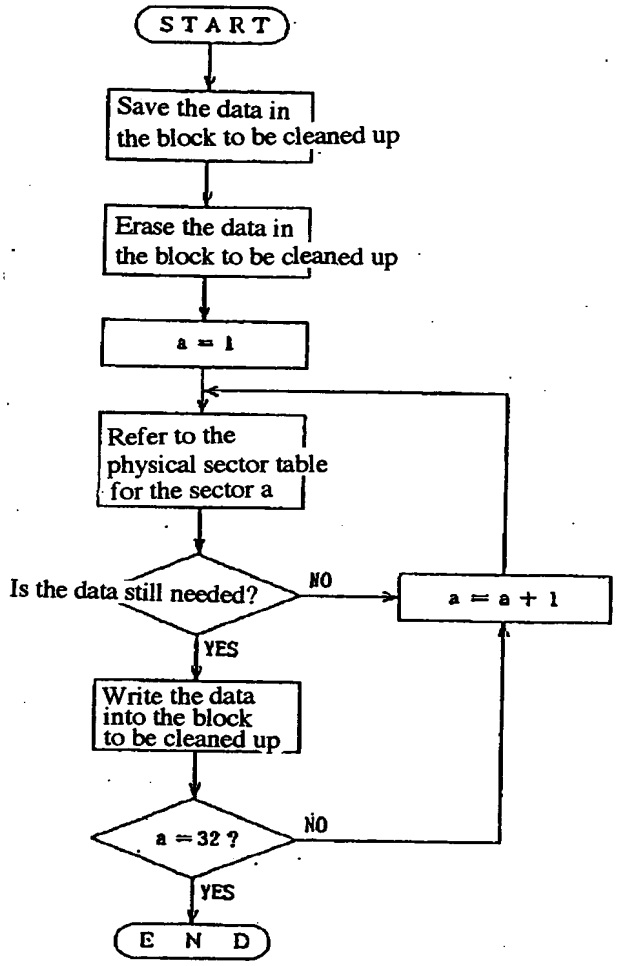

START
Save the data in the block to be cleaned up
Erase the data in the block to be cleaned up
a = 1
Refer to the physical sector table for the sector a
Is the data still needed?
NO
a = a + 1
YES
Write the data into the block to be cleaned up
a = 32 ?
NO
YES
E N D

Fig. 28

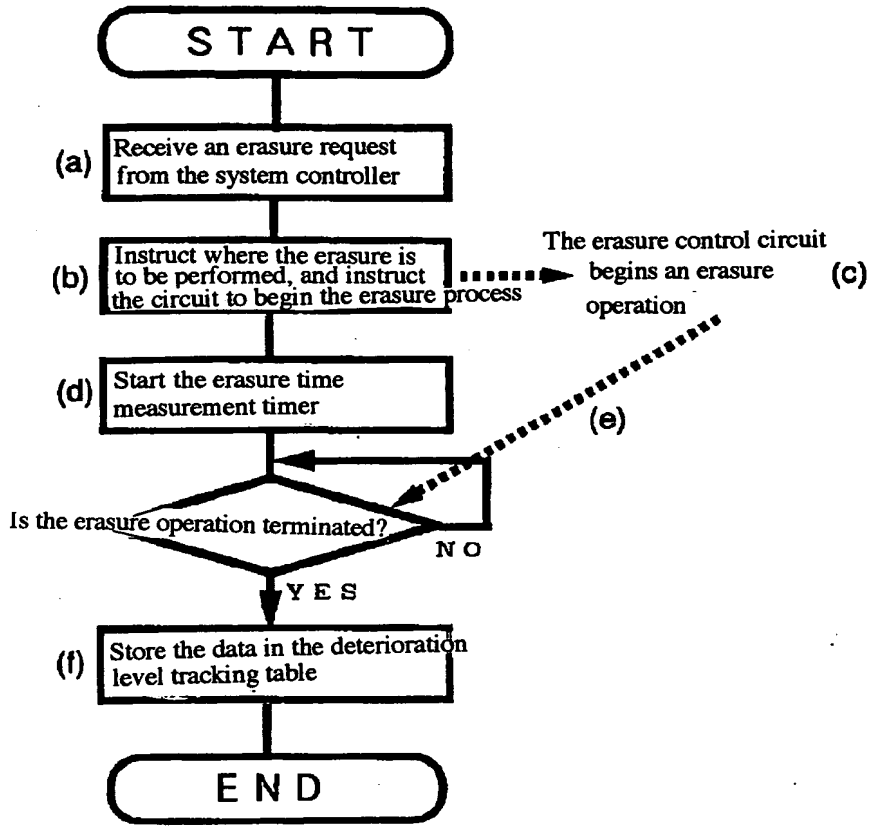

START
(a)
Receive an erasure request from the system controller
(b)
Instruct where the erasure is to be performed, and instruct the circuit to begin the erasure process
The erasure control circuit begins an erasure operation
(c)
(d)
Start the erasure time measurement timer
(e)
Is the erasure operation terminated?
NO
YES
(f)
Store the data in the deterioration level tracking table
END

Fig. 30

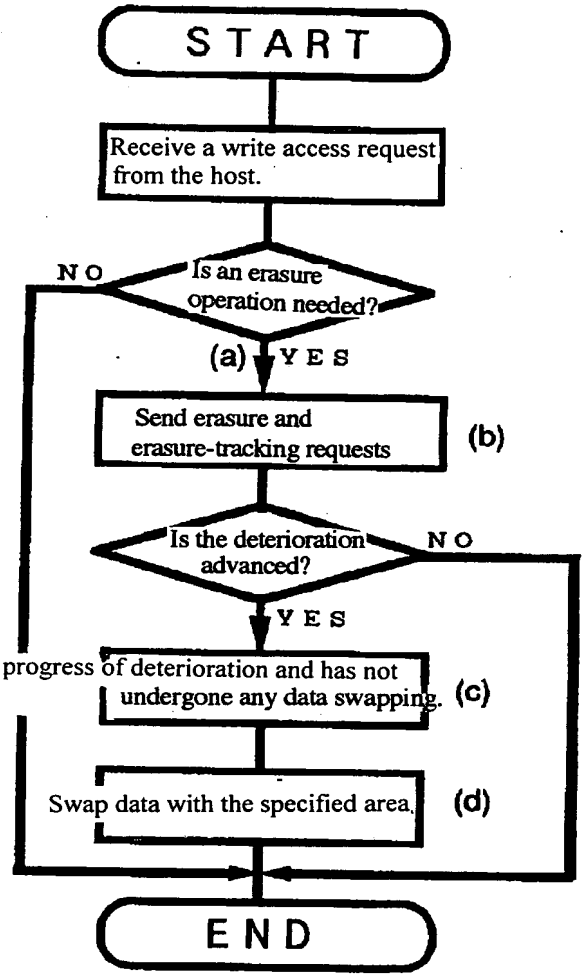

START
Receive a write access request from the host.
NO
Is an erasure operation needed?
(a)
YES
Send erasure and erasure-tracking requests
(b)
Is the deterioration advanced?
NO
YES
Find the area where there is least progress of deterioration and has not undergone any data swapping.
(c)
Swap data with the specified area.
(d)
END

Fig. 31

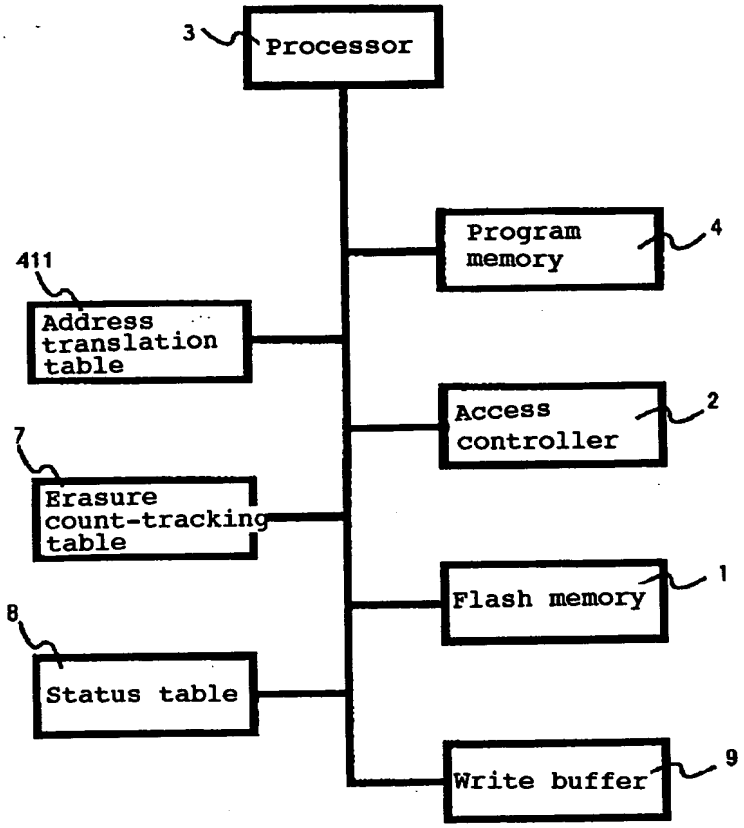

3
Processor
Program memory
4
411
Address translation table
Access controller
2
7
Erasure count-tracking table
Flash memory
1
8
Status table
Write buffer
9

Fig. 33

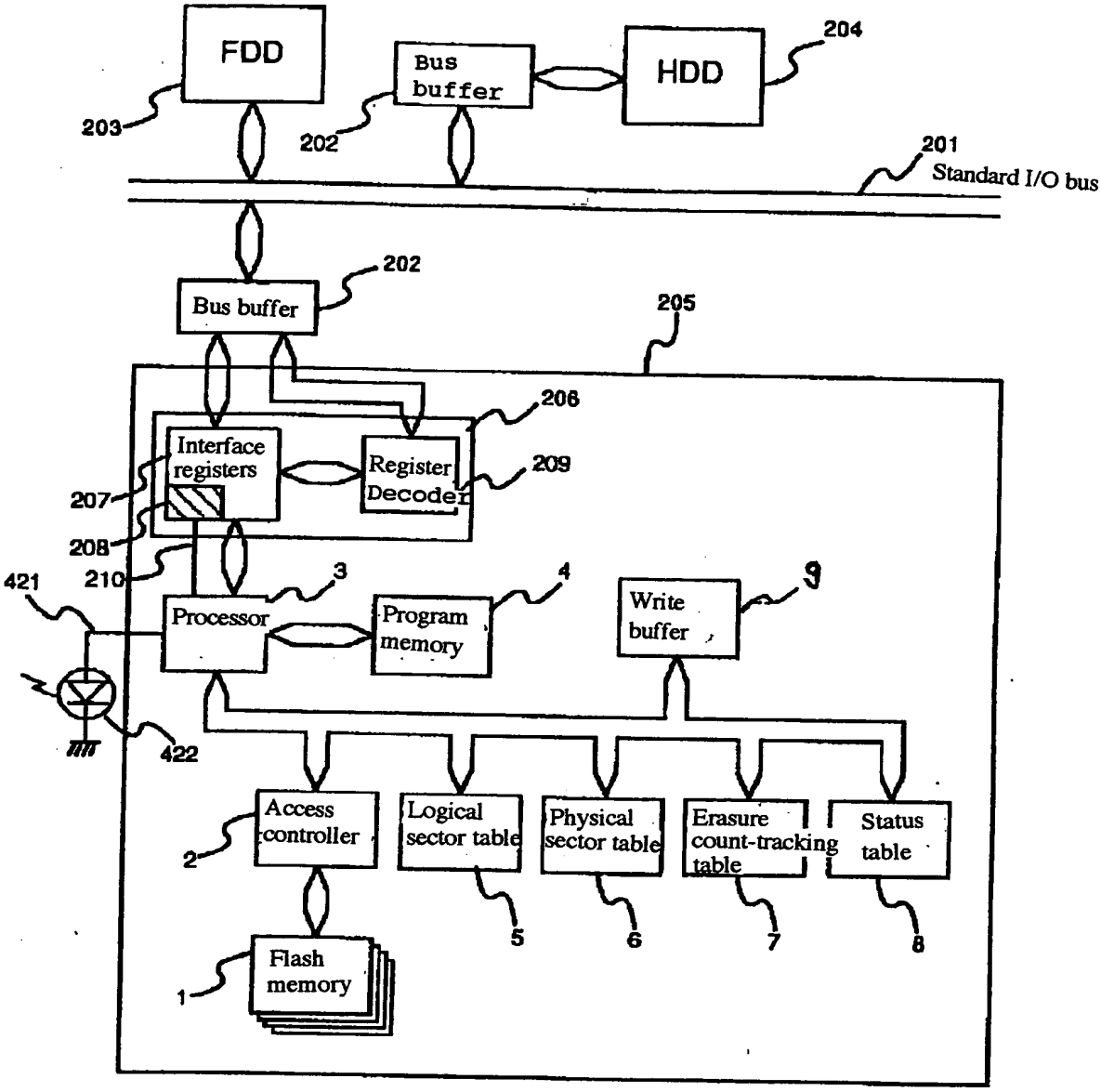

FDD
203
Bus buffer
202
HDD
204
201
Standard I/O bus
202
Bus buffer
205
206
Interface registers
207
208
210
Register Decoder
209
421
3
Processor
4
Program memory
9
Write buffer
422
Access controller
2
Logical sector table
5
Physical sector table
6
Erasure count-tracking table
7
Status table
8
Flash memory
1

Fig. 34

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開平6－124596
(43)公開日　平成６年(1994)５月６日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ１１Ｃ　16/06					
Ｇ０６Ｆ　12/16	３１０　Ｑ	7629－5B			
		6741－5L	Ｇ１１Ｃ　17／00	３０９　Ｃ	
		6741－5L		３０９　Ｆ	

審査請求　未請求　請求項の数41(全 36 頁)

(21)出願番号　特願平4－318159

(22)出願日　平成４年(1992)11月27日

(31)優先権主張番号　特願平3－314297
(32)優先日　平３(1991)11月28日
(33)優先権主張国　日本（ＪＰ）
(31)優先権主張番号　特願平4－31756
(32)優先日　平４(1992)２月19日
(33)優先権主張国　日本（ＪＰ）
(31)優先権主張番号　特願平4－230914
(32)優先日　平４(1992)８月31日
(33)優先権主張国　日本（ＪＰ）

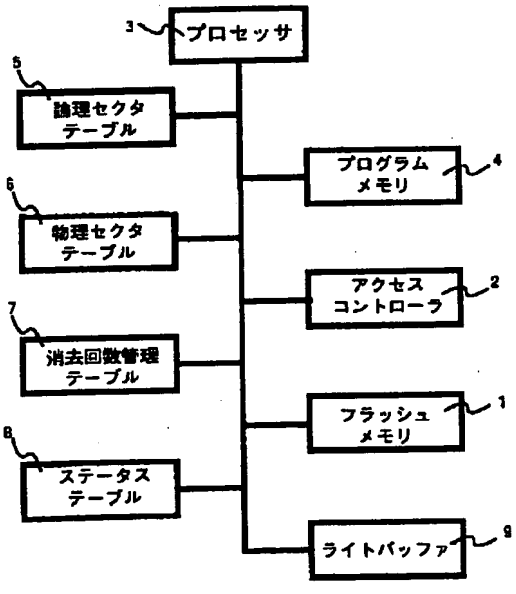
(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　片山　国弘
神奈川県横浜市戸塚区吉田町292番地株式会社日立製作所マイクロエレクトロニクス機器開発研究所内
(72)発明者　常広　隆司
神奈川県横浜市戸塚区吉田町292番地株式会社日立製作所マイクロエレクトロニクス機器開発研究所内
(74)代理人　弁理士　小川　勝男

最終頁に続く

(54)【発明の名称】　フラッシュメモリを使用した記憶装置

(57)【要約】
【目的】フラッシュメモリを用いて補助記憶装置を実現する際に、フラッシュメモリの欠点である消去回数の制限を補うハード構成とデータの扱い方に関する発明である。
【構成】フラッシュメモリ１内部を、ファイルデータの書き込み消去単位であるセクタに分け、各セクタに書き込んだファイルに番号を付し、これを管理するテーブル５、６を備え、すでに書いたファイルの書換え時にも新たなデータとして別のセクタに書き込む。そして古いデータは消去して新たに書き込み可能領域としてテーブル５、６上で管理する。
【効果】ファイル管理テーブルなどの頻繁に書換えがあるファイルを特定のセクタに割り当てないことにより、フラッシュメモリの消去箇所が一部に集中せずに分散し、フラッシュメモリの寿命を伸ばしシステム全体の信頼性を増す。

図1



1

【特許請求の範囲】

【請求項1】フラッシュメモリの記憶領域に番号を付した物理セクタと、記憶データを書き込むために用いる論理セクタ番号と実際に書き込むための物理セクタ番号とを対応付ける手段と、同一論理セクタ番号への書き込みにおいても消去回数に基づいて対応しない物理セクタに書き込みを行なうようにし該物理セクタへの書き込み回数が少数のセクタに集中しないよう制御する制御部とを有することを特徴としたフラッシュメモリを使用した記憶装置。

【請求項2】請求項1記載の記憶装置において、記憶領域に付した物理セクタ番号に対応した物理セクタ領域に書き込まれているデータの論理セクタ番号を登録した物理セクタ参照手段と、ある論理セクタのデータがどこの物理セクタに書き込まれているかを物理セクタ番号により登録した論理セクタ参照手段とを備えることを特徴としたフラッシュメモリを使用した記憶装置。

【請求項3】請求項1記載の記憶装置において、フラッシュメモリの最小消去単位毎に消去回数を記録した消去回数参照手段を備えることを特徴としたフラッシュメモリを使用した記憶装置。

【請求項4】請求項3記載の記憶装置において、データの書き換えを行う毎に消去管理テーブルの消去回数をチェックし、該消去回数がメモリの書き換え保証回数よりも小さくなるように設定した規定値の整数倍に達する毎に、消去回数が少ない記憶領域のデータを転送して書込むための消去回数管理手段を設け、消去回数の少ない記憶領域は別のデータを格納するような、消去回数の平均化を図ることを特徴としたフラッシュメモリを使用した記憶装置。

【請求項5】請求項1記載の記憶装置において、フラッシュメモリの最小消去単位の記憶領域ごとの状態を参照する状態参照手段を設けてあることを特徴としたフラッシュメモリを使用した記憶装置。

【請求項6】請求項1記載の記憶装置において、次の論理セクタの書き込みにおいてどの物理セクタに書き込むかを示す書き込みセクタポインタを設け、書き込みセクタポインタをカウントアップまたはカウントダウンしてメモリをアクセスするアドレス値として使用することにより書き込みセクタの順序に規則性をもたせることを特徴としたフラッシュメモリを使用した記憶装置。

【請求項7】請求項2記載の記憶装置において、該物理セクタ参照手段を不揮発性記憶手段で構成し、該論理セクタ参照手段は揮発性記憶手段で構成することとし、システムの電源投入時に初期設定処理として該不揮発性記憶手段に構成してある該物理セクタ参照手段の内容より該論理セクタ参照手段の内容を作り出して該揮発性記憶手段に格納することを特徴としたフラッシュメモリを使用した記憶装置。

【請求項8】請求項1記載の記憶装置において、フラッシュメモリの最小消去単位が物理セクタの記憶容量より大きい場合、該最小消去単位の記憶領域を複数の物理セクタに分割し、ある物理セクタの消去を行うために同一の最小消去単位の記憶領域に格納されている他の物理セクタのデータを一時的に退避するための記憶手段を備えたことを特徴とするフラッシュメモリを使用した記憶装置。

【請求項9】請求項8記載の記憶装置において、フラッシュメモリの書き込み時間の高速化を図るため、書き込みデータを一時的に格納する記憶手段を、請求項8記載のデータを一時的に退避するための記憶手段と共用することを特徴としたフラッシュメモリを使用した記憶装置

【請求項10】データの格納領域とは別に最小消去単位ごとに対応した情報を格納する記憶領域を設け、最小消去単位をアクセスするためのアドレスと同一アドレスにより該情報をアクセスできるとともに、データ格納の制御とは別にデータの出力を制御できるとを特徴としたフラッシュメモリチップ。

【請求項11】データの格納領域とは別に1セクタ以上のデータを格納するバッファ領域を備え、該バッファ領域には外部よりデータの書き込みが行え、かつ書き込んだデータを該データの格納領域に転送し書き込むことを特徴としたフラッシュメモリチップ。

【請求項12】データの格納領域とは別に1セクタ以上のデータを格納するバッファ領域を備え、該バッファ領域には前記データの格納領域のデータを1セクタ単位で転送でき、かつ転送した該バッファ領域上のデータを外部に取り出すことを特徴としたフラッシュメモリチップ。

【請求項13】データの格納領域とは別に最小消去単位毎に対応した情報を格納する記憶領域を設け、データの格納領域に不良の記憶領域が発生したときにはその場所を登録し訂正することを特徴としたフラッシュメモリチップ。

【請求項14】請求項1記載のフラッシュメモリを使用した記憶装置を補助記憶装置とする情報機器。

【請求項15】ＣＰＵと、主記憶装置と、初期設定及び基本処理プログラムを記憶するＢＩＯＳＲＯＭと、補助記憶装置として請求項1記載のフラッシュメモリを用いた記憶装置からなる情報機器において、該ＢＩＯＳＲＯＭに格納されているプログラムは補助記憶装置としてＨＤＤを制御するプログラムを含み、該制御プログラムにより該フラッシュメモリを用いた記憶装置を制御することを特徴とした情報機器。

【請求項16】請求項15記載の情報機器において、フラッシュメモリを用いた記憶装置がフラッシュメモリの素子劣化に伴う使用限界に達したことを判定する判定手段と、該判定手段による判定結果を記憶する記憶手段を備えたことを特徴とする情報機器。

【請求項17】請求項16記載の情報機器において、該

判定手段は、記憶装置の記憶媒体であるところのフラッシュメモリのうち少なくとも一つの消去動作が、予め定めた規定時間内に完了しなかったときに使用限界に達したと判定することを特徴とした情報機器。

【請求項１８】請求項１６記載の情報機器において、該判定手段は、記憶装置の記憶媒体であるところのフラッシュメモリの消去動作が、予め定めた規定時間内に消去動作が完了しなくなった記憶容量を累積し、総計が一定容量を超えたときに使用限界に達したと判定することを特徴とした情報機器。

【請求項１９】請求項１６記載の情報機器において、該判定手段は、記憶装置の記憶媒体であるところのフラッシュメモリの消去を行った回数を記録しておき、一つ以上のフラッシュメモリが予め定めた回数を越えてしまったときに使用限界に達したと判定することを特徴とした情報機器。

【請求項２０】請求項１６記載の情報機器において、該判定手段は、記憶装置の記憶媒体であるところのフラッシュメモリの消去を行った回数を記録しておき、予め定めた回数を越えてしまったフラッシュメモリの記憶容量が一定値を越えたときに使用限界に達したと判定することを特徴とした情報機器。

【請求項２１】請求項１記載のフラッシュメモリを用いた記憶装置を補助記憶装置と、中央処理装置と、表示装置からなる情報機器において、該記憶装置が記憶媒体であるフラッシュメモリの素子劣化による使用限界に達したことを判定する判定手段と、該判定手段による判定結果を記憶する記憶手段と、該判定手段により使用限界に達したと判断したときにこれを該中央処理装置に報告する手段を備えたことを特徴とする情報機器。

【請求項２２】請求項１６記載の情報機器において、該補助記憶装置が使用限界に達したと該判定手段が判定したときに、該判定を該中央処理装置に報告し、表示手段に使用限界に達したことを表示することにより使用者に警告することを特徴とした情報機器。

【請求項２３】請求項１記載のフラッシュメモリを用いた記憶装置を補助記憶装置と、中央処理装置と、警告音発生装置とからなる情報機器において、該記憶装置が記憶媒体であるフラッシュメモリの素子劣化による使用限界に達したことを判定する判定手段と、該判定手段による判定結果を記憶する記憶手段と、該判定手段により使用限界に達したと判断したときにこれを該中央処理装置に報告するとともに該警告音発生装置により使用者に警告することを特徴とした情報機器。

【請求項２４】請求項１６記載の情報機器において、該判定手段により判定されて該記憶手段に記憶されている判定結果を読み出す保守プログラムにより、該記憶装置が使用限界に達したことを使用者が認識できることを特徴とした情報機器

【請求項２５】請求項１５記載の情報機器において、フラッシュメモリを用いた記憶装置が処理を行っている際中に情報機器の電源を落された場合、フラッシュメモリを用いた記憶装置内に備えられた電池を駆動電源として処理を継続することを特徴とした情報機器。

【請求項２６】請求項１５記載の情報機器において、情報機器内の電源の制御をつかさどる電源制御装置と、フラッシュメモリを用いた記憶装置が処理途中であったらそれを該電源制御装置に伝達する伝達手段を備え、情報機器のデータやプログラムの演算処理をつかさどる中央処理装置部が動作を停止するとき、該電源制御装置は該伝達手段によりフラッシュメモリを用いた記憶装置の処理状況を調べて、もし処理途中であれば少なくともフラッシュメモリを用いた記憶装置への電源供給を続け、処理途中の処理を終了し電源の供給の必要がなくなったら電源供給を遮断することを特徴とした情報機器。

【請求項２７】フラッシュメモリに格納されているデータを一括消去するための最小単位である記憶領域ごとにメモリセルの劣化の度合いを診断する劣化診断手段と、該劣化診断手段による診断結果を記憶する劣化度記憶手段と、該劣化診断手段の診断結果と該劣化度記憶手段の記憶内容により該記憶領域に格納されている記憶内容を入れ換える記憶内容入換え手段を備えたことを特徴とするフラッシュメモリを使用した記憶装置。

【請求項２８】請求項２７記載のフラッシュメモリを使用した記憶装置の劣化診断手段において、メモリセルの劣化をフラッシュメモリの格納データを一括消去する最小単位となる記憶領域ごとの消去回数により診断することを特徴とするフラッシュメモリを使用した記憶装置。

【請求項２９】請求項２７記載のフラッシュメモリを使用した記憶装置の劣化診断手段において、メモリセルの劣化をフラッシュメモリの記憶内容の消去に要する時間により診断することを特徴とするフラッシュメモリを使用した記憶装置。

【請求項３０】請求項２８記載のフラッシュメモリを使用した記憶装置の劣化度記憶手段において、フラッシュメモリの一括消去の最小単位となる記憶領域ごとの消去回数を記憶することを特徴とするフラッシュメモリを使用した記憶装置。

【請求項３１】請求項３０記載のフラッシュメモリを使用した記憶装置の劣化度記憶手段において、記憶する消去回数を下位のビットを１ビット以上切り捨てて記憶し、劣化度の記憶容量を節約することを特徴とするフラッシュメモリを使用した記憶装置。

【請求項３２】請求項２７記載のフラッシュメモリを使用した記憶装置の劣化度記憶手段において、記憶する劣化度を、フラッシュメモリの記憶内容の消去に要する時間により２以上の段階に分け、その各段階に番号を対応して付し、その番号を記憶することを特徴とするフラッシュメモリを使用した記憶装置。

【請求項３３】フラッシュメモリの格納データを一括消

去する最小単位となる記憶領域ごとの消去回数によりメモリセルの劣化の診断をする劣化診断手段と、該消去回数を記憶する劣化度記憶手段と、該劣化診断手段により該消去回数があらかじめ規定された消去回数に達したことを確認したら、該劣化度記憶手段の記録内容から、劣化が起こりにくい記憶内容が格納されているために消去回数が少ないと判断される記憶領域を探し出し、該記憶領域の内容を該規定された消去回数に達した記憶領域に転送することを特徴とした記憶内容入換え手段を備えるフラッシュメモリを使用した記憶装置。

【請求項３４】フラッシュメモリの格納データを一括消去する最小単位となる記憶領域ごとの消去に費やす時間によりメモリセルの劣化を診断する劣化診断手段と、該消去に費やす時間により劣化の度合いを２以上の段階に分け、該段階に対応して番号を付し、これを記憶する劣化度記憶手段と、該劣化診断手段により該劣化度記憶手段に記憶されていた劣化の段階より劣化が１段階以上進んだと判断されたら、該劣化度記憶手段から劣化が低い段階にあって書き換えが起こりにくい記憶内容が格納されている記憶領域を探し出し、該記憶内容を前記劣化の段階が一つ以上進んだ記憶領域に転送することを特徴とした記憶内容入換え手段を備えるフラッシュメモリを使用した記憶装置。

【請求項３５】請求項２７記載のフラッシュメモリを使用した記憶装置において、記憶内容の入れ換えを行った場合には入れ換えた記憶領域に対するそれ以降のアクセスが正しく行えるように、各記憶領域に番地を付し、それぞれの入れ換えた先の記憶領域の番地を記憶して番地の変換が行える変換番地記憶手段と、その内容によりアクセスを正しく行う番地変換制御手段を備えたことを特徴とするフラッシュメモリを使用した記憶装置。

【請求項３６】フラッシュメモリを使用した補助記憶装置のホストとなる情報機器は、該補助記憶装置にデータの書き込みを要求すると、該補助記憶装置からの書き込み完了信号を受け取ることにより該書き込みデータの出力を終了し、該補助記憶装置においてはホストからのデータの受取りをフラッシュメモリ以外の別のメモリに対して行い、該フラッシュメモリ以外のメモリへのデータの受取りが終了した時点で該書き込み完了信号を該ホスト情報機器に出力することを特徴としたフラッシュメモリを使用した補助記憶装置。

【請求項３７】請求項３６記載のフラッシュメモリを使用した補助記憶装置において、該補助記憶装置のホストとなる情報機器からの書き込みデータをフラッシュメモリ以外のメモリへ格納し、これが終了して書き込み完了信号を該ホスト情報機器に出力した後で該フラッシュメモリ以外の別のメモリからフラッシュメモリへのデータ転送を行うことを特徴としたフラッシュメモリを使用した補助記憶装置。

【請求項３８】請求項３７記載のフラッシュメモリを使

用した補助記憶装置において、ホストとなる情報機器に対し、書き込み完了信号を出力した後に、フラッシュメモリ以外の別のメモリから該フラッシュメモリへのデータ転送を行っている間、処理中であることを使用者に知らせる発光手段を情報機器の筐体部に設けたことを特徴とするフラッシュメモリを使用した情報機器。

【請求項３９】請求項５記載の記憶装置において、参照すべき状態としては、該記憶領域が既書き込みデータで満たされているか、メモリセルが劣化して書き込み不可能な状態になってるかであることを特徴としたフラッシュメモリを使用した補助記憶装置。

【請求項４０】フラッシュメモリの記憶領域に番号を付した物理セクタが設けられており、記憶データの書き込みのために仮想的な番号を付した論理セクタを設け、該論理セクタへの書き込みにおいて論理セクタ番号と実際に書き込む物理セクタ番号を対応させず、同一論理セクタへの書き込みにおいても異なる物理セクタに書き込みを行ない、該物理セクタへの書き込み回数が少数のセクタに集中しないように書き込みを分散させたことを特徴としたフラッシュメモリへの記憶方法。

【請求項４１】分割された記憶領域を有するとともにファイルデータを記憶するフラッシュメモリと、各記憶領域の消去回数の累計を記憶する消去管理手段と、ある記憶領域に対して消去回数が少ない記憶領域のデータを転送して書き込ませるとともに、該消去回数が少ない記憶領域には消去回数が多い記憶領域のデータを転送して書き込ませるデータ制御手段とを有することを特徴とするフラッシュメモリを使用した記憶装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は携帯用の小型情報処理機器の補助記憶装置にかかり、特にフラッシュメモリを用いた半導体ファイル記憶装置及びそれを搭載する情報機器に関する。

【０００２】

【従来の技術】情報機器の補助記憶装置の従来技術としては磁気記憶装置が最も一般的であるが、磁気記憶装置では書き込むファイルをセクタと呼ぶ記憶単位に分割し、記憶媒体の物理的な位置に対応させて記憶する。すなわちあるファイルの書き換えにおいては基本的に同一位置に書き込みが行われ、書き込みデータが増えるとその分だけ新たなセクタへの書き込みを行う。これに対し別の補助記憶装置として光ディスク装置が挙げられる。現在一般的な光ディスクは、書き込みが一回だけ可能で消去は不可能である。従って一度書き込んだファイルの書き換え時は実際には書き換えを行わず、別の領域に書き込んで以前書き込んだデータは無効にして以後読み出さないようにする。つまり磁気ディスク装置と異なり、書き換えデータと記憶場所には全く関連性を持たせない、という方式で補助記憶装置の機能を果たしている。

以上のようなディスクを回転させて大容量のデータを高速にアクセスし、補助記憶装置の機能を果たす記憶装置に対し、半導体メモリを用いて補助記憶装置とする半導体ファイル記憶装置が近年脚光を浴びている。特に電気的に書き換えが可能な不揮発性メモリ（以下ＥＥＰＲＯＭと記す）を用いたものが今後半導体ファイル記憶装置の主流になると考えられる。それを実現する一つの技術として特開平３－２５７９８がある。これはＥＥＰＲＯＭを用いた記憶装置であり、ＥＥＰＲＯＭの欠点である書き換え消去回数の制限を保護し、ＥＥＰＲＯＭを用いて実用的な記憶装置を実現する方式である。その概要を説明すると複数のメモリ素子を用意し、各素子の消去書き換え回数を記録して管理し、ＥＥＰＲＯＭの書き換え保証回数より小さなある規定回数に達したら用意してあったメモリ素子に切り替えて使用することにより、記憶データの保護を図るものである。

【０００３】

【発明が解決しようとする課題】上記従来技術における光ディスク装置の方式ではファイルの書替えがあるたびに記憶領域をつぶしていくことになり、非常に大容量の記憶媒体がないと記憶領域の確保ができない、という点に問題がある。特にファイルの書替えが激しい情報機器の記憶装置とする場合は、実際の記憶容量がそれほど大きくなくても、記憶領域が大きくなってしまう。一方補助記憶装置として最も一般的な磁気ディスク装置の場合は、一度書き込んだファイルを書き替える場合、同じ領域に新しいデータを書き込む。しかしこれをＥＥＰＲＯＭに応用すると、記憶しているファイルの一覧となるファイル（一般にはディレクトリファイルと呼ばれる）やファイルの記憶場所を参照するためのファイル（ファイルアロケーションテーブル）等はデータのライトアクセスがあるたびに書替えが起こり、書替えが局所に集中する。これは書込み消去回数に制限のあるＥＥＰＲＯＭにおいては寿命を著しく短くすることになる。またＥＥＰＲＯＭを適用した特開平３－２５７９８においては、メモリ素子の劣化状態を消去回数により把握して壊れる前に代替のメモリに切り替えるという方式を発明しているが、この方式では実際の記憶容量の倍以上のメモリ容量を備える必要がある。すなわち代替のメモリは、最初に使用するメモリが壊れるまでは全く使用しないし、また壊れたメモリは壊れたあとは不必要になってしまう。これは物理的体積、重量に非常に無駄が大きくなる。しかも通常データの書き換えは全体的に起こる訳ではないため、部分的に劣化しているだけなのにメモリチップ全体を非使用状態にしてしまうのは経済的に無駄が大きいと言える。

【０００４】

【課題を解決するための手段】ＥＥＰＲＯＭの一種であるフラッシュメモリはデータの電気的消去が可能であり、従って不揮発性メモリでありながらデータの書き換えが可能である。そして消去単位が一般のＥＥＰＲＯＭと比較して大きなものであるため、セル構造が単純化され集積度を大きくすることができ、物理的にも経済的にも他のメモリ素子よりも大容量の補助記憶装置に適している。そこで欠点である消去回数の制限を保護すべく消去の回数が極力少なくなる方法でデータの記憶と書き換えを行うことにより他の補助記憶装置に置き換えて使用することが可能となる。その手段としてデータの書き込みは光ディスクと同様に記憶データと記憶場所には関連性は持たせず、データの書き込みがあったらデータを書き加えていくこととし、既に書き込んであるファイルの書き換えが発生した場合は、古いファイルの記憶領域を無効として消去可能領域にする。そしてあるタイミングをもって無効領域のデータを消去するガーベイジコレクションを行う。ガーベイジコレクションは、消去単位が大きく無効領域のデータとともに消去の必要がない有効領域のデータを消去しなければならない時には、有効領域のデータを別の記憶領域に移し、元の場所を消去して新たな書き込み可能領域にする。さらに消去回数を管理する消去回数管理テーブルを設け、ガーベイジコレクションを行うごとに回数をインクリメントする。そしてあるブロックが規定した消去回数に達したら消去回数の少ないブロックとのデータの交換を行う。

【０００５】

【作用】上記手段によれば光デイスク装置のようにデータの書替えが起こるたびに記憶領域をつぶしていくことはなく、不必要になったデータを消去して記憶領域とすることができ、また磁気ディスクのように書替えが局所に集中してフラッシュメモリの寿命を縮めることはない。たとえ書き込み領域が著しく小さくなり、さらに書き替えるデータも非常に限られたものとなって、消去ブロックが特定のブロックに集中しても、消去回数の多いブロックのデータと消去回数の少ないブロックのデータとを入れ替えることにより、頻繁に書替えが起こる領域を固定せず、また書替えがあまり起こらない領域を固定せずに全記憶領域を一律に使用する記憶方式を実現する。従ってメモリの劣化が鈍化し、代替メモリを備える必要がなくなる。

【０００６】

【実施例】以下に本発明の実施例を述べる。まず第１の実施例を図１、図２、図３及び図４により説明する。図１は第１の実施例を実現するためのハードウエア構成であり、図２は本実施例に用いるフラッシュメモリチップの内部構成を示した図であり、図３は格納データ管理のメインルーチンのフローチャートであり、図４は消去管理ルーチンのフローチャートである。

【０００７】最初に図２によりフラッシュメモリの動作について述べる。図中１１はメモリチップ全体、１２はデータ書き込み単位、１３はデータ消去の最小単位で消去ブロックと呼ぶこととする。フラッシュメモリは電気

9

的消去可能なＰＲＯＭであり、一種のＥＥＰＲＯＭであるが、ノーマルなＥＥＰＲＯＭがデータの書き込み単位と消去（書換え）単位が同一であるのに対し、フラッシュメモリは書き込み単位より消去単位が極めて大きく、一度書き込んだデータを書き換えるためには他の多くのデータも同時に消去しなければならない。その代りの利点としてノーマルなＥＥＰＲＯＭより集積度を高くでき、大容量の記憶装置に適している。図２においてメモリチップ全体１１が一つ以上の消去ブロック１３に分割されており、書き込み単位１２が１ワード（メモリにおけるワードでありメモリ構成に従う）であるのに対し、消去ブロック１３はそれより大きい領域となる。フラッシュメモリを補助記憶装置として用いる際には、磁気ディスク装置の仕様に合わせ、５１２バイトを消去単位とすると使いやすくなる。

【０００８】次にこのメモリを用いた記憶装置の第１の実施例のシステム構成とその動作を図１及び図３、図４を用いて説明する。なお以下の説明ではファイルデータの記憶単位をセクタと呼ぶこととし、本実施例では１セクタが１消去ブロックと一致するものとする。図中、１はファイルデータの記憶をするフラッシュメモリチップ群、２はフラッシュメモリ１のアクセス信号を生成するアクセスコントローラ、３は記憶データやステータスデータを操作してフラッシュメモリによる外部記憶システムを構築するマイクロプロセッサ、４はプロセッサ３を動作させる制御プログラムを格納したプログラムメモリ、５はある論理アドレスで示されるセクタ（論理セクタ）のデータがフラッシュメモリ１上のどこにマッピングされているかを参照するための論理セクタテーブル、６はフラッシュメモリ１上の物理的なアドレスで示される物理セクタにマッピングされたファイルデータの論理セクタ番号を参照するための物理セクタテーブル、７は各物理セクタの消去回数の累計を記録する消去回数管理テーブル、８は各物理セクタのステータスを参照するステータステーブル、９はデータの書き込みを高速化するために書き込みデータを一時的に保存するライトバッファである。次に動作を説明する。本システムに対しデータのアクセス要求をするシステム（ホストシステム）よりデータのリードアクセス要求があると、プロセッサ３は論理セクタテーブル５を参照して該当する論理セクタが格納されている物理セクタを割り出し、その物理セクタをアクセスして要求されたデータをホストシステムに送出する。ホストシステムからのデータの書き込み要求に対しては、図３のフローチャートを用いて説明する。図３のフローチャートはプログラムメモリ４に格納されているプログラムのメインルーチンのフローチャートでありその流れを説明すると、まず、次のデータの書き込みをするセクタを示す書き込みポインタが設定されており、このポインタが示すセクタが書き込み可能な状態にあるかをステータステーブル８により判別する（ａ）。

10

ステータステーブルでは消去回数が多くなり劣化して使いものにならなくなったことを示すフラグやすでにデータが書き込まれていることをポすフラグがあり、これらが立っていて書き込み不可能であれば次のセクタにポインタを移す（ｂ）。そして書き込み可能であればそのセクタへのデータの書き込みを行う（ｃ）。そして既に一度書き込んだことのある論理セクタの書き換えである場合は、前回書き込んだ当該論理セクタのデータはもはや不要であるため、論理セクタテーブルより不要になったデータが書き込まれている物理セクタを捜し出してこれを消去し、同時に前回の書き込みの際に書いた物理セクタテーブル６の内容を消去する（ｄ）。セクタの消去が行われたら消去管理ルーチンへ飛ぶ。消去管理ルーチンについては後述する。そして論理セクタテーブル５に書き込みポインタの示す物理セクタ番号を、物理セクタテーブル６には書き込みポインタの示す場所に書き込んだ論理セクタ番号を書き込む（ｅ）。なお（ｂ）においてはデータが書き込まれているかの判断をステータステーブル８に書き込むことにしても良いし、物理セクタテーブル６により判断することもできる。物理セクタテーブル６は未書き込みセクタである場合には全ビットＨにするかあるいはＬにすれば判別が容易になる。次に先述の消去管理ルーチンについて図４を用いて説明する。まずデータを消去した物理セクタに対応する消去管理テーブルの消去回数カウンタを１インクリメントし（ａ）、消去回数が規定回数に達していなければメインルーチンに復帰し、規定回数に達していたら、データの入替えを行う（ｂ）。データの入替えを行うためにはまず他の全セクタの消去管理テーブルを調べ、消去回数が最小でかつまだデータの入替えを行っていないセクタを捜しだす（ｃ）。捜し当てた消去回数が最小のセクタに格納されているデータを、先程消去を行ったセクタに書き込む（ｄ）。書き込みを行ったら両セクタのステータステーブルの入替えフラグを立てる（ｅ）。この入替えフラグは、消去回数の少ない物理セクタが、本ルーチンにより消去が頻繁に起こるようになるのに、それを示す手段がないと再び消去回数が少ないセクタに選ばれてしまうことが考えられ、この時にデータを入れ替えた消去回数の多いセクタは、また消去が頻繁に起こるようになってしまう、ということが起こるのを防ぐものである。この入替えフラグを立てたら、該当する論理セクタテーブルの内容と物理セクタテーブルの内容を書き換える（ｆ）。以上が終了したらメインルーチンに復帰する。なお消去回数が最小として選ばれたセクタは一度消去されることになるため、消去管理テーブルの消去回数カウンタをインクリメントする必要がある。また入替えフラグは全てのセクタのフラグが立ったらクリアされるか、あるいは論理の判断を反転する。つまり１のとき入替えが行われたと判断していたものを０になったら入替えが行われたことにする。また（ｂ）の規定回数は、フラッシュメモ

リの書換え可能回数の保証値より小さな値の倍数とすべきで、例えば１００００回の保証回数であれば、１００００回の倍数回や、２０００回、５０００回の倍数回等が適当である。このルーチンにより、限られたブロックに消去が頻繁に起きたら消去回数の少ないブロックのデータと入替えをして、消去回数の多いブロックに消去があまり起きないセクタのデータを格納することにより、消去回数の平均化を図るものである。これは通常の補助記憶装置の格納データには大変効果があると考えられる。例えばオペレーションシステムプログラムを格納した領域ではデータの書換えは全く起きないが、アプリケーションプログラムのデータとなるグラフィックデータやテキストデータの領域では頻繁にデータの書換えが起きる。そのため消去回数の平均化を行わないと、システムプログラム領域のメモリはデータが変化しようがないため、消去回数の増加による劣化は全く起きないことになり、それ以外のデータ領域のメモリは限られたメモリ空間で頻繁に消去が行われ、消去回数を急増させることになる。つまり使用可能領域が少ない状態で特に大きな効果があるといえる。

【０００９】以上が第１の実施例の動作の説明である。本実施例によればプロセッサを搭載したことによりプログラムメモリの内容に従った細かな制御ができるようになり、またライトバッファにより書き込みの高速化が図れ、ステータステーブルを備えたことにより各セクタの状態を記録するのに拡張性がある。そしてフラッシュメモリの寿命を延ばすために、消去回数を管理した最適なファイル管理が行える効果がある。

【００１０】次に第２の実施例について図５、図６、図７、図８及び図９を用いて説明する。図５は第２の実施例におけるハードウエア構成であり、図６は本実施例におけるフラッシュメモリ内の記憶構成を示した図であり、図７はデータ書き込みのためのメインルーチンのフローチャート、図８は不必要なデータを格納したセクタを未書き込みセクタにするための整理ルーチンのフローチャート、図９は消去回数の管理をする消去管理ルーチンのフローチャートである。

【００１１】まず図６により本実施例におけるフラッシュメモリのチップとその使用法を説明する。図中、５２は第１の実施例で用いたセクタと呼ぶファイルデータを記憶する記憶領域の単位である。５３はデータ消去の最小単位で消去ブロックと呼び、複数のセクタ５２により構成される。すなわち第１の実施例とは異なり消去ブロック５３とセクタ５２の記憶容量は異なるものとする。５４はメモリチップ全体であり、図では複数の消去ブロックにより構成されているが、１チップに１消去ブロックであることも考えられる。本実施例ではセクタ単位にファイルデータの格納を行うが、そのファイルデータの書換えの要求があって消去するためには他のセクタも同時に消去されてしまうようなメモリチップに適応するものである。図５は本実施例のハードウエア構成で、図中、４１は書き込みデータの記憶領域であるフラッシュメモリで一つ一つはメモリチップ５４である。４２はフラッシュメモリ４１をアクセスするアクセスコントローラ、４３は記憶データやステータスデータを操作するプロセッサ、４４はプロセッサ３を動かすための制御プログラムが格納されているプログラムメモリ、４５はフラッシュメモリ４１の物理セクタ５１が記憶している論理セクタ番号を参照するための物理セクタテーブル、４６は記憶してある論理セクタ番号のデータがフラッシュメモリ１のどこの物理セクタに記憶されているかを参照するために物理セクタ番号が記録されている論理セクタテーブル、４７は各ブロックの消去回数を記録する消去管理テーブル、４８は各ブロックのステータスを記憶するステータステーブル、４９は既書き込みセクタ数を参照するための書き込みセクタ数テーブル、５０は書き込みの高速化を図るため、書き込みデータを一時的に保持するライトバッファ、５１は整理ルーチンを行う際に用いる整理バッファである。

【００１２】次に動作を説明する。リードアクセス時は第１の実施例同様、論理セクタテーブル４５と物理セクタテーブル４６を参照して行う。一方ライトアクセスは、図７を用いて説明すると、まず書き込みポインタを設定し、その書き込みポインタで示されたブロックのステータステーブルを参照する（ａ）。すなわち本実施例における書き込みポインタはブロック単位のポインタである。そしてこわれていたら次のブロックにポインタを移す（ｂ）。こわれていなければそのブロックの書き込みセクタ数テーブル４９を参照する（ｃ）。そして既にブロック内の全セクタが書き込み済であったら整理ルーチンに飛ぶ。整理ルーチンについては後述する。もし未書き込みセクタがあったらデータの書き込みを行い（ｄ）、該当する書き込みセクタ数テーブルを１インクリメントする（ｅ）。従って、例えば図６において一つ前の書き込みを第１ブロックの第３物理セクタに書き込んだ場合、次の書き込みは第１ブロックの第４物理セクタに書き込みを行うことになる。実際のライトアクセス制御はアクセスコントローラ２により行う。ここで書き込みを行ったブロックのセクタが全て書き込まれてしまったら整理ルーチンに飛ぶ（ｆ）。そして書き込み後は物理セクタテーブルおよび論理セクタテーブルに書き込んだセクタ番号をそれぞれ記録する（ｇ）。もし以前に書き込んでいる論理セクタの再書き込みであれば、以前物理セクタテーブルに書き込んだ論理セクタ番号を消去する。これはその物理セクタのデータは無効であることを示すための動作である。なお（ｆ）の動作は整理時間短縮のために設けられておりフラッシュメモリの消去効率を高くするためには行うべきでない。次に整理ルーチンについて説明すると、整理ルーチンとは書き込みポインタの指し示すブロックがすでに全セクタ書き込まれて

13

おり、書き込み不可能である時の動作ルーチンである。整理ルーチンが必要となる理由は、これまで説明した書き込み方法では同じファイルのデータの書き替えにおいて別の物理セクタへ書き込みを行う。すなわち書き換える前に書き込んだデータは不要となるがメモリ上には記憶されたままであり消去すべきデータである。しかし不要となるたびに消去していたのでは消去回数が有限であるフラッシュメモリにとっては寿命を短くすることになる。そこで整理ルーチンによって消去する。整理ルーチンはブロックが書き込みデータで一杯になったときに行う。整理の具体的な方法は、図８のフローチャートに従って行う。以下フローチャート内の各部の説明をする。整理ブロックの物理セクタテーブルを参照し、ブロック内に消去可能なセクタすなわち他の物理セクタに新たに書き換えられたため不必要となったセクタがないかをチェックする（a）。消去可能なセクタが一つもなかったらメインルーチンに復帰し、一つでもあったら整理を行う。まず整理するブロック内のデータを整理バッファ５１に退避し（b）、整理するブロックの消去を行う（c）。消去を行ったら図９の消去管理ルーチンに飛ぶがこれについては後で説明する（d）。そして整理バッファの中で必要なセクタだけを整理するブロックに復帰させる（e）。復帰したときのセクタの位置は、元々の場所に復帰させれば論理セクタテーブル４５や物理セクタテーブル４６を書き換える必要がない。また復帰の際にはそのブロックの若いセクタ番号から埋めていく方法を取ると、次の新しいセクタの書き込みがし易くなるが、論理セクタテーブル４５と物理セクタテーブル４６を書き換える必要がある。いずれの場合も書き込みセクタ数テーブルは復帰したセクタ数に書き換えることになる（f）。次に消去管理ルーチンについて説明する。図９は消去管理ルーチンのフローチャートであり、基本的には第１の実施例で説明したものと同様であるが、消去の管理がセクタではなくブロック単位に行う点が異なる。まず消去が行われたブロックの消去回数が一定数に達したか、をチェックし（a）、もし達していなければこのルーチンを脱出。達していたら（b）全ブロックの消去回数を検索し、消去回数が最小のブロックを捜しだす（c）。整理を行ったブロックと消去回数が最小のブロックが一致していなければこの２つのブロックのデータを入れ替える（d）。入替えには図５の整理データバッファ５１を利用する。そして入替えを行ったブロックの入替えフラグを立てる（e）。消去回数カウンタをインクリメントするとともに論理セクタテーブル及び物理セクタテーブルを書き換える（f）。以上が本実施例における消去管理ルーチンの動作である。第１の実施例同様このルーチンにより、限られたブロックに消去が頻繁に起きたら消去回数の少ないブロックのデータと入替えをして、消去回数の平均化を図るものである。

【００１３】以上が第２の実施例の動作説明である。本

14

実施例によれば、消去ブロックが書き込むセクタの単位としては大き過ぎるときに、消去ブロックを分割して効率良く使える効果がある。

【００１４】次に第３の実施例を図１０、図１１及び図１２を用いて説明する。本実施例においてはメモリチップは第１の実施例と同様の図２の消去ブロックとセクタの容量が一致するものとするが、使用時は図１０に示す構成をとる。図中９１は複数のセクタよりなるブロックである。本実施例ではセクタと消去ブロックの単位が同一であるため本実施例のブロック９１は複数の消去ブロックで構成されることになる。既出の他の番号は図２あるいは図６と同様のものである。ハードウエア構成は図１１に示したものであり図中１０１はブロック９１ごとの消去回数を記録した消去管理テーブルであり、詳細は後述するが結果的にブロック９１内の消去回数の累計を記憶することになる。他は図１、図５と同様のものである。図１２は本実施例の消去管理ルーチンのフローチャートであり、メインルーチンは図３と同様である。メインルーチンの動作は第１の実施例の説明に従うとして、メインルーチンから消去管理ルーチンに飛んでからの動作を図１２により説明する。まず消去を行ったセクタを含むブロックの消去管理テーブルの回数カウンタを１インクリメント（a）。消去回数が規定値に達したかを判別し（b）、達していたら前ブロックの消去回数を調べて回数が最小で、かつまだデータの入替えを行っていないブロックを割り出し（c）、２つのブロックのデータを入替える（d）。そして入替えフラグを立てるとともに（e）、各テーブルの内容を書き換える（f）。本実施例の特徴はセクタ単位の消去ができるメモリにおいて消去管理を複数のセクタで行うことにより、消去管理の簡便化を図って、大きなファイルの書換えにおける待ち時間の節約と、消去管理テーブルの削減ができるという効果が期待できる。従って大容量の記憶装置で、消去管理をセクタごとに行うことが困難な場合に適応する。

【００１５】ところでこれまで説明した実施例で構成要素となっていた論理セクタテーブル、物理セクタテーブルなどのテーブルについて説明を加える。フラッシュメモリは不揮発性のメモリであるため、当然のことながら本発明のシステムは非動作時には電源の供給を停止してもフラッシュメモリ内のデータが失われることはない。しかしながらいくらデータが保存されていても、テーブルの内容が失われると、どのセクタに何のデータが格納されているかがわからなくなり、フラッシュメモリ内のデータは正体不明の無意味なデータと化してしまう。そのためテーブルに格納されているデータも電源供給停止後に保存されている必要がある。ただしテーブル内の全てのデータを保存する必要はない。例えば論理セクタテーブルのデータは物理セクタテーブルのデータから容易に作成可能である。逆もまた可能であるのでつまりどちらか一方のデータが保存されていれば良いことになる。

15

そこで物理セクタテーブルを電気的消去可能で書き込み可能な不揮発性メモリ（ＥＥＰＲＯＭ）に記憶し、論理セクタテーブルは電源供給を開始するシステムの起動時に論理セクタテーブルより作成することとする。こうすると論理セクタテーブルは揮発性のメモリを使用できる。またこれと同様に各ブロックの使用セクタ数テーブルも物理セクタテーブルより作成可能であるため、システムの起動時に揮発性メモリに作成する。これらのテーブルはメインシステムの主メモリ上に展開することも可能である。その他のテーブルについては、消去管理テーブル、ステータステーブルがあるが、これらは他のテーブルからの作成は不可能であり、消去管理テーブルはデータが失われるべきでないためＥＥＰＲＯＭに記憶する。ステータステーブルはもう一度書き込みや消去をしたときに得られるが、二度手間になるためＥＥＰＲＯＭに記憶すべきである。しかしメモリの節約のため揮発性メモリに記憶することもできる。これらテーブルの記憶媒体としては同じメモリに格納してチップ数を減らすこともできる。例えば物理セクタテーブルと消去回数テーブルはどちらも不揮発性メモリに格納すべきであるため、同じＥＥＰＲＯＭチップに格納してＥＥＰＲＯＭは１チップだけにすることができる。またプロセッサをワンチップマイコンとした場合は、使用セクタ数テーブルなどの比較的小さなテーブルはワンチップマイコンに内蔵されているＲＡＭコアに格納する。これらをまとめた第４の実施例の構成図を図１３に示した。図中、１１１はＲＡＭ、ＲＯＭを内蔵したワンチップマイコン、１１２はワンチップマイコン内のＲＡＭコア、１１３はワンチップマイコン内のＲＯＭコア、１１４はＥＥＰＲＯＭチップ、１１５はＳＲＡＭまたはＤＲＡＭ、以下既出の番号は、前述と同様のものである。ＲＡＭコア１１２には使用セクタ数テーブルを格納し、ＲＯＭコア１１３にはマイコンの制御プログラムを格納し、ＥＥＰＲＯＭ１１４には物理セクタテーブル及び消去管理テーブルを格納し、ＲＡＭ１１５には論理セクタテーブルを格納する。またＲＡＭ１１５の空き領域では図９で示した整理ルーチンを行う際の整理データバッファ、及び書き込みの高速化を図るライトバッファとしても用いることとする。このように本実施例によれば記憶媒体の特徴にあわせ、テーブル、バッファ類をまとめてチップ数の削減を図ることができる。これに対し、図２４はＥＥＰＲＯＭを省略した構成となる実施例である。これは第４の実施例で説明した不揮発性のテーブルデータをフラッシュメモリ１に格納することにより実現する。図中、１１６はフラッシュメモリ１内に設けたテーブル格納領域である。ただしフラッシュメモリはテーブルデータのような小さい単位のデータの更新には適さないため、フラッシュメモリ１にテーブルのデータを格納するのは電源遮断の直前とし、通常の使用状態においては揮発性のＤＲＡＭやＳＲＡＭ１１５に格納するものとする。つまり本発

16

明のシステムの電源を遮断する際に、ＲＡＭ１１５の内容のうち必要なデータをフラッシュメモリ１に転送してから電源を遮断し、次に電源を投入した時にフラッシュメモリ１からＲＡＭ１１５にデータをロードしてから通常の動作を開始する。従ってフラッシュメモリにはテーブル格納用の記憶領域１１６を常に確保しておく必要がある。ただしこの格納位置を固定とする必要はない。固定としなかった場合はあらかじめテーブルを格納するアドレスを示す領域を設けておき、テーブルを格納したアドレスを常に記録しておくことにより、電源立ち上げ時にテーブルの位置をサーチする必要がなくなる。

【００１６】またさらにフラッシュメモリ自体の構成における実施例を図１４、図１５、図１６及び図２３に示した。図１４はフラッシュメモリチップ内に消去ブロック単位にデータ領域と異なるテーブル用の記憶領域を持たせたものであり、これに各ブロックごとの物理セクタテーブルや消去回数を書き込むことによりＥＥＰＲＯＭを省略できる。図中、１３１は消去ブロック、１３２は消去ブロック１３１ごとに付加されるテーブル用の記憶領域である。消去ブロック１３１に対応するテーブル１３２に格納されたデータは消去ブロック１３１と同一のアドレスでアクセス可能とし、信号線による選択やモードの選択によりファイルデータとテーブルのデータを出し分ける。このようにするとデータ領域とテーブル領域の寿命が一致し、データ領域１３１が使用可能なのに、テーブルがこわれて使用不能に陥るということがなくなる。データ領域とテーブル領域が接近していればその傾向がさらに強くなる。そして消去ブロックごとにテーブルを構成するためアドレスラインの共有化が簡単に実現する。テーブル用の記憶領域１３２の一例を図２３（ａ）に示した。図中、１３３は消去単位１ブロック全体を表し、１３４はブロック内のファイルデータ領域、１３５はそのブロックの論理番号を格納する領域で８ビットの記憶容量を持つ。１３６はそのブロックの状態を示すステータステーブルで１６ビットの容量である。ステータステーブル１３６の内容は、消去回数管理テーブル、使用不能フラグ、入替え済みフラグ、訂正フラグ等が挙げられる。このステータステーブル１３６で余ったビットをブロック番号格納に用いることもできる。図２３（ｂ）はさらにエラー訂正領域を持たせたテーブル領域１３２の一例であり、１３７はエラー訂正個所を示す領域である。これは訂正能力に応じて記憶容量を設定することとする。たとえば１ワード１６ビットのワード構成で、消去ブロックが５１２Ｂの構成のメモリチップに対してエラー訂正領域が８ビットであれば、１ワード訂正能力を持つ。つまり不良ビットの存在するワード番号をここに書き込んでおき、訂正データを他の領域に書き込んでおけば、読み出しの際にそのワードのデータを置き換えてしまうことにより実現する。訂正データ領域をこのテーブル内に持たせてもよいし、他の記憶領域にま

とめておく方法もある。前者ではデータ訂正をテーブル内で全てまかなうことができるが1ブロックごとに訂正データ領域を持つこととなるためチップ全体で冗長領域が大きくなってしまう。後者では訂正データが書き込まれている領域を示す手段が必要になり、また他の記憶領域をアクセスするためアクセス時間が大きくなってしまうが、訂正データ数に応じて冗長記憶領域を大きくも小さくも設定できる。なお訂正データが書き込まれている領域の指定はステータステーブル領域１３６の余りビットを使用すると良い。また訂正データ領域の一例として図１５を用いて説明する。図１５はデータの記憶領域とは別に1バイトや1ワード等の細かい単位で書き込み消去ができるＥＥＰＲＯＭタイプの記憶領域１３８を持たせたものであり、この領域を訂正データテーブルとして、訂正データを書き込んでおき訂正すべき個所のデータの読み出しの際に指定された位置のデータを読み出して置き換えることによりデータ訂正が実現する。また図１５は、テーブル領域の構成を、対応するデータ領域の消去ブロックの付近に構成せず、一まとめにして同様の効果を狙った実施例と考えることもできる。この場合図内の構成要素は図１４と同様であり、メモリセルの構成がチップ内で一まとめになるため、図１４のメモリセルの構成より簡略化される効果がある。

【００１７】また図１６は別の構成のフラッシュメモリチップである。データの記憶領域１４１とは別に、データを一時的に保持するバッファ領域１４２をメモリチップ内に持たせたものであり、この部分は揮発性のメモリでもよい。１４３はクロック入力によりカウントアップするアドレスカウンタである。ライトアクセス時はライトデータをバッファ領域１４２に書き込み、アドレスを入力することにより複数のデータを一気にデータ領域１４１に転送し書き込めるようにする。このようなメモリを用いれば書き込みの高速化を図る外付けのライトバッファを省略できる。またリード時にも逆にデータ領域１４１からアドレスを入力することにより複数のデータを一度にバッファ領域１４２に転送できれば、リードアクセスも簡略化される。この場合バッファはシリアルアクセスメモリとして連続アドレスの入力が必要でなく、内部にアドレスカウンタ１４３を備え、クロックを入力すれば内部のアドレスカウンタがカウントアップし、連続した領域をアクセスしてデータを出力できるようにすればさらに使い勝手が良くなる。なおバッファ領域はセクタを1単位として構成するのが最も効果的であり、1単位分に限らず複数単位分構成するとバッファ効果を向上させることができる。例えば1セクタが消去ブロック単位であった場合、1セクタのデータを格納するバッファが1つ備えられていれば、1セクタの書き込み読み出しが一度に行われるようになるが、複数備わっていれば複数のセクタのデータの書き込みを受け入れることができ、また読み出しデータを用意することができる。そしてこのバッファを利用すれば外付けの整理バッファも省略できる効果がある。

【００１８】次にこれまでの実施例で説明してきたフラッシュメモリを用いた記憶装置を情報処理システムに応用する実施例について説明する。図１７はフラッシュメモリを用いた記憶装置（以下、フラッシュファイルシステムと称す）を情報処理システム（以下ホストと称す）と接続するためのインタフェース回路を説明する図であり、図中２０１はホストの外部Ｉ／Ｏバスであり、標準的なバスとしてはＩＳＡ、ＥＩＳＡ、マイクロチャネル、ＳＣＳＩなどのバスが挙げられる。２０２は標準バスを専用バスに変換するためのバスバッファあるいはバスコントローラであるが、これが省略されるシステムも考えられる。これらに接続されているのはホスト側が自システム内の主記憶装置あるいは拡張主記憶装置、表示記憶装置等の記憶装置に記憶しきれないデータや電源遮断後も保持したいデータなどを記憶するために設置している外部記憶装置あるいは補助記憶装置である。フロッピディスクドライブ２０３、ハードディスクドライブ２０４等が一般的であるが、それに加えフラッシュファイルシステム２０５が接続されている。なおこれら全ての補助記憶装置がホストシステムに接続される必要はなく、ユーザが適宜選択して接続するものである。フラッシュファイルシステム２０５にはインタフェース回路２０６が付加されており、２０７はインタフェースレジスタ群、２０８はインタフェースレジスタ群の中のレジスタの一つであるコマンドレジスタ、２０９はインタフェースレジスタ群のアドレスデコード回路、２１０はコマンド割込み信号である。以下既出の番号はこれまでの説明で述べてきたものと同様のものである。ただしプログラムメモリ４に格納されているプログラムはこれまでの説明で述べてきたファイルデータの制御、管理に加え、ホストからのアクセス要求を中心としたコマンドへの対応のプログラムが格納されている。ホストはホストバス２０１を通して補助記憶装置にコマンドを出す。これはインタフェース回路２０６内のインタフェースレジスタ群２０７の一つであるコマンドレジスタ２０８にコマンドコードを書き込むことによって行なわれる。インタフェースレジスタ群２０７は、ハードディスクドライブがインタフェースレジスタとして持つレジスタを全て備え、またレジスタの仕様も一致し、ホストからはハードディスクをアクセスするのと何ら変わらないようにしている。なおこのレジスタをフロッピディスクドライブや光ディスクドライブ等、他の補助記憶装置のインタフェースに合わせることも有効であると考える。あるいは複数の補助記憶装置のインタフェースを同時にサポートし、ホストからは別の補助記憶装置を利用しているようにみえるが、実際には1台のフラッシュファイルシステムでまかなうというのはスペース的に非常に有効である。さてコマンドレジスタ２０８はホストによりコマン

ドを書き込まれると割込み信号２１０をプロセッサ３に対して発し、これを受けたプロセッサ３は書き込まれたコマンドコードを解釈して、ホストのコマンド要求に応える。なおインタフェースレジスタ群２０７やコマンドは全てハードディスクドライブに対応するものであるが、記憶媒体が異なるものであるため、不必要なものや処理が異なるものもある。例えばフォーマットは磁気ディスク装置には不可欠のものであるが、半導体ディスクドライブでは不必要であるため、特に処理を行なわないようにしたり、単に規則的なデータに書き換えたりといった処理にする。以下、ファイルデータのリードライトに関してはこれまでの実施例で説明したものと同様の動作とする。なお図は第１の実施例を適用しているが、これまで説明したあるいはこの後説明する他の実施例にそのまま適用することも可能である。

【００１９】図１８はフラッシュファイルシステムを補助記憶装置とするパーソナルコンピュータの一例の構成図である。図中、２２１は本情報機器のＣＰＵ、２２２はコプロセッサ、２２３は本実施例の情報処理システム内部に構築した標準I/Oバス、２２４は標準I/Oバス２２３を構築するバスユニット、２２５は主メモリや拡張メモリなど高速メモリをアクセスするメモリ制御ユニット、２２６は主メモリ、２２７は基本制御プログラムが格納されたＢＩＯＳ ＲＯＭ、２２８はキーボードコントローラでこの先にはキーボードが接続されているものとする。２２９は表示アダプタでこの先には何らかの表示装置が接続されているものとする。２３０は拡張メモリ、２３１はプリンタなどを接続するパラレルポートI/F、２３２はマウスやＲＳ２３２Ｃなどのシリアルポートの/F、２３３はフロッピディスクドライブ、２３４は標準I/Oバス２２３より標準のＨＤＤI/Fに変換するバッファコントローラ、２３５はフラッシュファイルシステムである。このフラッシュファイルシステム２３５の内部は、これまでの実施例で示したものにより構成されており、２３６はファイルアクセスを受け付けてデータの受渡しを行なうI/Fユニットであり、図１７におけるインタフェースレジスタ群２０７及びアドレスデコード２０９にあたる。２３７はフラッシュファイルシステム２３５内部のデータ管理や制御を行なうコントロールユニットであり、図１７におけるプロセッサ３及びプログラムメモリ４、アクセスコントローラ２、ライトバッファ９よりなる。２３８はファイルデータを格納するフラッシュメモリアレイ、２３９はデータ及びメモリ管理のための情報を記憶するインフォメーションテーブルであり、図１７におけるテーブル５、６、７、８を全て含むものである。次に動作を説明する。電源が投入されて動作を開始するとまずＣＰＵ２２１はＢＩＯＳ ＲＯＭ２２７を標準I/Oバス２２３を通してアクセスし、初期診断、初期設定を行なう。そして補助記憶装置からシステムプログラムを主メモリ２２６にロードする。本実施例

では補助記憶装置としてフラッシュファイルシステム２３５を採用している。ただしＣＰＵ２２１は標準I/Oバス２２３を通してＨＤＤコントローラ２３４にＨＤＤをアクセスするものとして動作する。従ってフラッシュファイルシステム２３５は内部のI/Fユニット２３６によってＨＤＤ完全互換のI/F機能をサポートしている。システムプログラムのロードが終了すると、ユーザの処理要求に従い、処理を進めていく。なおユーザは標準I/Oバス２２３上のＫＢＤＣ２２８や表示アダプタ２２９により処理の入出力を行ないながら作業を進める。そして必要に応じてパラレルI/F２３１、シリアルI/F２３２に接続された入出力装置を活用する。また本体上の主メモリ２２６では主記憶容量が不足する場合は、拡張ＲＡＭ２３０により主記憶を補う。ユーザがファイルを読み書きしたい場合にはユーザはＨＤＤが補助記憶装置であるものとして補助記憶装置へのアクセスを要求し、フラッシュファイルシステム２３５はそれを受けてファイルデータのアクセスを行なう。本実施例によれば標準的なパーソナルコンピュータの標準的なＩ／Ｏバス上に、現在最も一般的な補助記憶装置であるＨＤＤを搭載しているかのように、フラッシュファイルシステムを採用している。従って新しい記憶媒体としてフラッシュファイルシステムを採用してもＢＩＯＳ ＲＯＭやシステムプログラムをIIDDを搭載していたものに対し変更を加えなくてもそのまま使用できる。なお上記実施例は標準的なパーソナルコンピュータを例にしており、主メモリや拡張メモリが標準Ｉ／Ｏバス上にあったり、コプロセッサやパラレルI/F、シリアルI/Fが存在しない構成の情報機器も考えられこれらに関しても本発明の適用が可能である。

【００２０】次にフラッシュメモリの不良発生に対処する実施例について図１９を用いて説明する。フラッシュメモリは原理的に書き換え回数に限界があることはすでに述べているが、書き換えによる劣化が進むとデータ消去書き込みの消費時間が長くなり、またデータの保持信頼性に問題が出てくる。そこでメモリの劣化により使用すべきでない状態に達した、と判断されたときにはその記憶領域あるいはメモリチップの使用を中止すべきである。そしてそのような領域が増え、フラッシュファイルシステム全体の寿命が近づいたと判断されたときには、そのフラッシュファイルシステムの使用を中止すべきである。フラッシュメモリの劣化を検出する方法としては、消去を繰返し行っても規定回数以内で消去が完全にならない、あるいは消去時間がある規定時間以上かかってしまう、という消去不良から判断する方法。そして書き込みを繰返し行っても規定回数以内で書き込みが完全にならない、という書き込み不良から判断する方法。また消去回数を管理して規定回数以上に達してしまったことから判断する方法が考えられる。そして劣化して使用不能に陥った記憶容量がある規定値に達したところでフ

ラッシュファイルシステムとしての寿命と判断する。図１９はそれをユーザに警告するための手段を実現する一実施例であり、図中２４０はフラッシュファイルシステム２３５内のコントローラが自システムの使用限界を認識し、これをホストシステムに伝えるための割込み信号であり、２４１は使用限界に達したことを示すエラー報告レジスタである。ホストはＨＤＤバッファ２３４を通して割込み信号２４０を受けるとフラッシュファイルシステム内のレジスタ２４１をアクセスして状況を把握し、適宜ユーザに報告する。図２０はこの動作を実現するソフトウェアをフローチャートで示したものであり、（1）はフラッシュファイルシステム内でのチェックルーチンのフローチャート、（2）はホストシステムでのユーザへの警告プログラムのフローチャートである。図の説明を兼ねながら動作説明を行なう。フラッシュファイルシステムのコントローラは使用限界に達したと見られる記憶領域が検出できたらこのルーチンにジャンプしてくる。そして使用限界の記憶領域の容量を加算していく（a）。そしてある限界値に達した場合（b）には、自システム内のエラー報告レジスタにエラー内容として使用限界に達したことを示す値を書き込む（c）。そしてホストへの割込み信号を出力して（d）、本ルーチンを終了する。なお（a）の容量の加算値はフラッシュメモリ内あるいは他のメモリに保持しておく必要がある。また（b）の残り容量の限界値はシステムに応じて適宜設定するものとする。さて（d）による割込み信号を受けたホストシステムでは、区切りの良いところで処理を中断し（2）のフローチャートで示したルーチンに入り、まずフラッシュファイルシステム内のエラー報告レジスタをアクセスし割込みを受けた理由を判別し（e）、これがフラッシュファイルシステムの使用限界に達したための使用中止要求であったら（f）、それ以後のフラッシュファイルシステムのアクセスを禁止し（g）、ユーザに警告する（h）。警告の方法としては、情報機器の表示装置を利用したり警告インジケータを取り付けるなど視覚的に訴える方法と、警告音を発するなどの聴覚的に訴える方法が考えられる。また、書き込みアクセスだけを禁止しリードアクセスは許可するようにするとファイルデータのバックアップがとれるようになる。本実施例によればフラッシュファイルシステムが使用限界に達したらすぐにユーザに知らせることができ、データの信頼性を増すことができる。別の実施例として図１８における割込み信号２４０は設けずに、ホストシステムがアクセスの度にエラー報告レジスタを読み出すこととして、図２０（2）のルーチンを実行する。本実施例によればホストシステムのハードウェア構成を従来と全く変えずに、ＨＤＤをそのまま置き換えて、使用限界報告ができる。また別の実施例としては、定期的に保守プログラムをホストシステムで実行し、その際にエラー報告レジスタ２４１を読み出して、使用限界に達していないかをチェックすることにより、使用限界を認識する。本実施例によればホストシステムのＢＩＯＳプログラムやシステムプログラムに全く変更を加える必要がない。

【００２１】次に電源遮断時の処理に対応する実施例を示す。本発明のフラッシュファイルシステムは基本的に補助記憶装置であり、ホストシステムより電源を供給される構成が一般的である。しかし動作的にはホストとは非同期であるため、ホストが非動作中にもフラッシュファイルシステムが動作している場合もある。しかしホストを扱うユーザがそれを認識していないとホストの電源を落してしまい、動作中のフラッシュファイルシステムへの電源供給が停止してしまう、ということが起こり得る。そこでこれに対応する発明の実施例を図２１に示す。図２１はホストからの電源供給が停止した後も、フラッシュファイルシステムが動作を続けられるようバッテリでバックアップ電源を供給する構成のシステムの図であり、図中、２５１はホストとなるパーソナルコンピュータ等の情報機器、２５２はフラッシュファイルシステム、２５３はバックアップ用のバッテリ、２５４はホスト２５１の電源が遮断されたときに、バッテリ２５３の電力がホスト２５１に逆流しないための逆流遮断回路である。２５５はホスト２５１からの電源供給が絶たれたことを検出する検出回路、２５６はその出力である検出信号、２５７はバッテリ２５３のフラッシュファイルシステム２５２への電力供給を遮断する遮断回路、２５８はフラッシュファイルシステム２５２が処理途中であった処理を終了し、電源供給を必要としなくなったため遮断回路２５７を動作させる遮断信号である。ホスト２５１から電源が供給されている間は、ホスト２５１の電源によりフラッシュファイルシステムが駆動される。これはバッテリ２５３の出力電圧とホスト２５１の供給電圧をうまく調整することによりバッテリ２５３にトリクル充電を施すことができる。ただしバッテリ２５３が２次電池でない場合には流入防止のダイオードなどを付加する必要がある。そしてホスト２５１の電源が落されてフラッシュファイルシステム２５２への電力供給が停止されると、バッテリ２５３から電源が供給されてフラッシュファイルは動作を続けることができる。そして電源遮断検出回路２５５が電源遮断を検出すると、検出信号２５６によりフラッシュファイルシステムがこれを認識し、処理中の作業を完了させ、その後電源遮断処理を行う。電源遮断処理では最後に遮断信号２５８によりバッテリ遮断回路２５７を遮断させて、フラッシュファイルシステムの動作を終了する。なお逆流遮断回路２５４は逆流防止用ダイオードを用いる方法が挙げられる。本実施例によればホスト２５１はフラッシュファイルシステムと信号上は完全に分離しているため、ホストのアクセス要求によるデータのやりとりだけとなり、ホストの動作停止はホスト内で閉じることができる。つまりホスト

はフラッシュファイルシステムを接続したことによるハード上の変更の必要がない。

【００２２】図２２はホストの電源遮断に対応する別の実施例である。ただし図２２の例では実際にはホストの電源による供給を続ける方式である。図中、２５８はホスト２５１からフラッシュファイルシステム２５２に供給される電源ライン、２５９はフラッシュファイルシステム２５２が作業中であり、電源供給の必要があることを示す電源ビジー信号である。ホスト２５１の電源回路は電源スイッチがＯＦＦになっても電源ビジー信号２５９がアクティブの間は電源供給を停止せずに、フラッシュファイルシステムの処理が全て終了して電源ビジー信号２５９がインアクティブになったら電源供給を停止する。つまりユーザにとってはホストの情報機器の電源スイッチを切り、電源を遮断してしまったものとした時でも実動作上は遮断されずに、フラッシュファイルシステム２５２の作業終了を待機する。本実施例によればフラッシュファイルシステムの動作終了をホストが認識して電源を制御することにより、バックアップ電源などが必要なくフラッシュファイルシステムの電源回路の構成がシンプルになる効果がある。

【００２３】次に本発明の第５の実施例を述べる。図２５は本発明を実現するためのハードウエア構成であり、図２６は本発明のフラッシュメモリ内の記憶構成を示した図であり、図２７はデータ書き込みのためのフローチャートである。便宜上図２６より説明する。図中、３１１は５１２バイト（４キロビット）を単位とする記憶領域で、実際にデータを書き込むセクタであるため物理セクタと呼ぶ。これに対し記憶データの供給側は、論理セクタと呼ぶ仮想のセクタによりデータを管理することとする。全ての物理セクタ３１１および論理セクタにはそれぞれ相関性の全くない番号を付す。３１２はデータを一括消去する単位となる消去ブロックであり、その容量はメモリにより異なる。例えば消去ブロック単位が１６キロバイトであれば３２物理セクタが１ブロックとなる。３１３はメモリチップであり、消去単位１６キロバイトで４メガビットのチップであれば３２ブロック１０２４セクタに区分けされる。チップ単位でしか消去できないものは１チップ１ブロック、物理セクタごとに消去できるものはブロック数と物理セクタ数は同一である。フラッシュメモリは消去が行われたブロックに対しては、そのブロック内においてランダムライトアクセスが可能である。ただし一回書き込んだら次の消去を行わなければ書き込みはできない。従って複数セクタを１ブロックとするメモリにおいては、各セクタが全て書き込まれてから消去すべきである。以下の説明においてはフラッシュメモリは４メガビットのチップで、消去ブロックが１６キロバイト（１２８キロビット）のものを使用することとする。図２６はまさにその構成のメモリを図示している。図２５はこのメモリチップ３１３を用いた場

合のハードウエア構成で、図中３０１は書き込みデータの記憶領域であるフラッシュメモリ、３０２はフラッシュメモリ３０１をアクセスするアクセスコントローラ、３０３は記憶データやステータスデータを操作するプロセッサ、３０４は記憶してある論理セクタ番号のデータがフラッシュメモリ３０１のどこの物理セクタに記憶されているかを参照するために物理セクタ番号が記録されている論理セクタテーブル、３０５はフラッシュメモリ３０１の物理セクタ３１１が記憶している論理セクタ番号を参照するための物理セクタテーブル、３０６は各ブロックのステータスを記憶するブロックテーブルで、そのブロックが使用可能であるか、あるいはそのブロックには何セクタ書き込まれているか、などが記録される。３０７はフラッシュメモリは書き込み速度が読みだしに比べ非常に遅いため、書き込みデータを一時的に保持してデータの供給側に速くバス権を復帰させるためのライトバッファである。３０８は後述の整理ルーチンにおいて整理データを一時的に保持する整理データバッファである。図２７は図２５の構成において書き込みを行うための図２５のプロセッサ３０３の動作を説明するフローチャートである。以下このフローチャートに従って動作を説明する。書き込むデータを記憶する際は１セクタ＝５１２バイト単位により行う。すなわち５１２バイトに満たないデータも５１２バイトの記憶領域に格納される。その１セクタ単位のデータに番号を付し、これを論理セクタ番号とする。データの供給側はこの論理セクタ番号だけを意識すればよいものとする。今、データ供給側が１セクタの書き込みを要求したとすると、このデータに、ある論理セクタ番号を付し、以後このデータは書替えが起こっても先の論理セクタ番号で扱われる。一方データの記憶側ではこのデータの格納場所を決定する。格納場所は書き込みポインタに指し示されたブロックの未書き込みの最初のセクタである。つまり一つ前の書き込みをした物理セクタの次のセクタである。例えば図２６において一つ前の書き込みを第１ブロックの第３物理セクタに書き込んだ場合、次の書き込みは第１ブロックの第４物理セクタに書き込みを行うものとする。この際のデータの格納場所の判断をプロセッサ３０３がブロックテーブルをアクセスして行う。この作業がフロチャート内の(a)である。（b）は書き込みに問題ない場合はそのブロックの書き込みセクタ数をインクリメントしてデータを書き込む。実際のライトアクセス制御はアクセスコントローラ３０２により行う。そして書き込み後は物理セクタテーブルおよび論理セクタテーブルに書き込んだセクタ番号をそれぞれ記録する。また以前に書き込んでいる論理セクタの再書き込みであれば、物理セクタテーブルに以前書き込んだ論理セクタ番号を消去する。これはその物理セクタのデータは無効であることを示すための動作である。（c）の整理ルーチンとは書き込みポインタの指し示すブロックがすでに全セクタ書き込ま

れており、書き込み不可能である時の動作ルーチンである。これは後述の図２８のフローチャートにより説明する。（ｄ）はブロックがこわれていたり、整理ルーチンに入った場合には書き込みが不可能であるため、そのブロックへの書き込みを中止し、次のブロックへの書き込みを行うための動作である。一方リードアクセスの際は必要なデータが格納されている論理セクタ番号で論理セクタテーブルを参照し、物理セクタ番号を割り出して必要なデータを読み出す。次に整理ルーチンについて説明する。整理ルーチンが必要となる理由は、これまで説明した書き込み方法では同じファイルのデータの書き替えにおいて別の物理セクタへ書き込みを行う。すなわち書き替える前に書き込んだデータは不要となるがメモリ上には記憶されたままであり消去すべきデータである。しかし不要となるたびに消去していたのでは消去回数が有限であるフラッシュメモリにとっては寿命を短くすることになる。そこで整理ルーチンによって消去する。整理ルーチンはブロックが書き込みデータで一杯になったときに行う。整理の具体的な方法は、図２８のフローチャートに従って行う。以下フローチャート内の各部の説明をする。（ａ）整理するブロック内の全データを一度図２５の整理データバッファ３０８に退避する。（ｂ）退避し消去可能になったところでそのブロックを消去する。（ｃ）整理データバッファにある各セクタの物理セクタテーブルを参照して、データが必要か不必要かを判断する。（ｄ）必要なデータであれば再びそのブロックに書き込む。以上が本発明の一実施例の動作説明である。本実施例によればフラッシュメモリの各消去ブロックの消去が一部に集中せずに順番に行われるため、寿命が延びるという効果が期待できる。また書き込みにおいてフラッシュメモリの書き込み速度が遅いという欠点を補う効果がある。また不必要なデータを効率的に消去するため、半導体ディスクとして常に記憶容量を保っていられる効果がある。

【００２４】次に他の実施例を説明する。図２９はフラッシュメモリの劣化を判定する装置の１構成例であり、図中４０１はフラッシュメモリセル、４０２は消去制御の開始から終了するまでの時間を測定する消去時間測定タイマ、４０３はフラッシュメモリセル４０１に書き込まれているデータを消去するための制御を行う消去制御回路、４０４はフラッシュメモリセル４０１の劣化度を記憶する劣化度管理テーブル、４０５はこれらを統括制御するコントローラ、４０６は劣化管理コントローラ４０５が消去制御回路４０３に消去を開始させる際に消去時間測定タイマ４０２を同時に起動するための起動信号、４０７は消去制御回路４０３が消去処理の終了を消去時間測定タイマ４０２に知らせる終了信号、４０８は消去時間測定タイマが測定した測定データである。図３０は図２９のコントローラ４０５の動作を説明するフローチャートであり、フラッシュメモリの劣化の判定をこ

のフローチャートに従って行う。次に動作を図２９、図３０を用いて説明する。まずシステム全体の制御からあるメモリセル４０１に消去の動作が必要になったとき、コントローラ４０５が消去要求を受ける（図３０ａ）。するとコントローラ４０５は消去制御回路４０３に消去箇所と消去動作の開始を指示する（図３０ｂ）。これを受けた消去制御回路４０３は該当するメモリセル４０１の消去動作を開始する（図３０ｃ）。これと同時にコントローラ４０５は消去時間測定タイマ４０２を起動する（図３０ｄ）。コントローラ４０５は消去終了まで待機し、メモリ制御回路４０３がメモリセル４０１の消去を完了すると、コントローラ４０５にそれを伝える（図３０ｅ）。コントローラ４０５は消去時間測定タイマ４０２を参照して消去に費やした時間を認識し、劣化の度合いを判別してそれに応じた番号をその記憶領域に付す。例えば劣化の度合いを８段階に分け、未使用状態と同様な消去時間であれば"０"、劣化が進んで使用不可能な状態を"７"としその間の段階を"１"から"６"の番号をあてる。そしてその記憶領域に付した番号を劣化度管理テーブル４０４に格納する（図３０ｆ）。劣化度を８段階にすることにより、一つの記憶領域を１バイトに割り当てられ、管理がし易くなる。各記憶領域の扱いは、この劣化度管理テーブルをもとにする。劣化度が一つ進んでしまった領域はその都度劣化の小さい領域のデータを転送して劣化の進行をとめ、全記憶領域の劣化の平均化を図る。また劣化が最終段階に達した領域は使用を禁止する。これは記憶システム全体を統括して制御するコントローラが行う処理である。この記憶システムのコントローラの動作フローを図３１に示した。同図を追って順に説明すると、ホストシステムから記憶システムにライトアクセス要求があり、データの書き込みを行う際に記憶システムのコントローラが消去動作を必要とすると判断したら（図３１ａ）、図２９のコントローラ４０５に消去と劣化管理の要求を出す（図３１ｂ）。消去動作が終了すると、消去を行った記憶領域の劣化度を図２９の劣化度管理テーブル４０４により参照し（図３１ｂ）、劣化が進んでいた場合は劣化が最も進んでおらず、かつ入替えをまだ行っていない領域を探し出して（図３１ｃ）、データを転送して入替えを行う（図３１ｄ）。この劣化が最も進んでおらず、かつ入替えをまだ行っていないという判断は、第１の実施例で説明した入換えフラグを利用した方が効率的である。これは一度上記入替え処理を行った領域は、格納されているデータが劣化の進まないデータ（書換えが起きにくいデータ）ではないため、劣化平均化の入換えに使用すべきではなく、入換えフラグを立てて明示する。なお本実施例では記憶システム全体の制御と劣化判定の装置における制御を分け、コントローラを別のものにしているが、コントローラ４０５を記憶システムのコントローラと共用しても良い。また消去時間測定タイマ４０２もハードウェアとして構成

されているが、コントローラによるソフト制御による消去時間測定でもよい。記憶システムの部品点数削減を目指すならこれらはなすべきことである。本実施例によれば劣化管理のためのテーブルをこれまでの実施例で採用していた消去管理テーブルに変えることができ、使用する記憶領域を飛躍的に節約することができる。しかもメモリの劣化という判定においては消去回数よりも消去時間の方がより直接的な判定材料となるため、より正確な劣化度の把握と、劣化の平均化が図れる効果がある。

【００２５】次に図２４の実施例を利用した消去回数管理の別の実施例を図３２により説明する。これまでの実施例における消去回数管理の方法は、消去が行われると１回ずつカウントアップして消去回数を細かく把握していたが、本実施例では消去回数管理テーブルの記憶容量削減のため下位のビットを切り捨ててしまう。図２４の実施例では通常の使用状態においては消去回数をＳＲＡＭやＤＲＡＭなどの揮発性メモリに最下位ビットまで格納し、電源遮断時にフラッシュメモリに転送することとしていたが、この転送時に下位のビットから１ビット以上を切り捨てる（図３２ａ）。あるいは桁上げして転送する（図３２ｂ）。なお捨ててしまう下位のビットは、消去回数の把握による劣化の認識が確度を失わない程度として、フラッシュメモリの消去回数の制限値により最適値を考えるべきであり、ビット切捨てにより切り捨てられてしまう消去回数の最大値がフラッシュメモリの消去回数の保証値の１％を越えない程度とする。つまり消去回数の保証値が１万回であれば、１００回を越えない６ビット（最大値６３）を切捨ての限度とする。本実施例によれば消去回数の把握が正確でなくなるが、誤差はフラッシュメモリの消去回数の保証値の１％以下としたため、もともと保証値の確度が一桁程度の幅があることから、消去が起きる傾向が強いデータか否かを判断するものであると考えれば、本質的な問題とはならない。これにより消去回数の管理に使用するフラッシュメモリの容量を削減することができる効果は大きい。

【００２６】次にテーブル領域を削減するための他の実施例を説明する。図３３はこれまでの実施例で説明している物理セクタテーブル及び論理セクタテーブルを省略した記憶システムの構成図である。ただしこれらの代わりになるアドレス変換テーブル４１１を備える。他の既出の番号はアドレス変換テーブル４１１は劣化度平均化のためのデータ入替えを行っていない記憶領域のアドレス入力に対しては変換を行わず、データ入替えを行っている記憶領域のアドレス入力に対しては、入れ替えた先のアドレスを出力するテーブルである。従って物理セクタテーブルや論理セクタテーブルを必要とせず、基本的にはシステムからアクセス要求のあったアドレスをそのままメモリ上の物理的なアドレスに対応してアクセスし、その領域の劣化が進んで劣化度の平均化のためのデータの入替えを行ったら、アドレス変換の対象となる。

本実施例は第１に実施例のような、ファイルデータの記憶単位（セクタ）と消去単位が一致している場合にだけ適用できる。そしてアドレス変換テーブルの容量は変換可能にできる領域の容量に依存する。すなわちアドレス変換テーブルの容量を小さくすれば変換を行える記憶容量が小さくなり、大きな領域を確保すれば多くの領域のアドレスを変換できるようになる。これはシステムの寿命に直接関与する要因である。なお図３３は第１の実施例に改良を加えた形となっているが、消去管理テーブル７の代わりに図２９により説明した劣化度管理テーブルを備えた実施例にも適用できる。本実施例によれば、データ管理が簡素化され、またテーブル領域を大きく削減できる効果がある。

【００２７】次に動作中を示すインジケータを設ける実施例を説明する。図３４は本実施例を示した内部構成例を示した図であり、図１７をもとにしており既出の番号は図１７と同様のものである。その他図中４２１はライトバッファ９からフラッシュメモリ１へのデータの転送中を示すプロセッサ３の出力ポート信号、４２２は出力ポート信号４２１により転送中であることを発光により示すインジケータである。インジケータとしては発光ダイオードが適当である。図３５は本実施例の外観図であり、（１）はカード形状の全体図、（２）は使用中の例を示した図である。図中４２３は本発明の補助記憶装置であるＩＣカード、４２４はコネクタ、４２５は発光ダイオード、４２６はホストパソコンである。図３４において標準ＩＯバス２０１によりホストのパソコンが補助記憶装置２０５に書き込みアクセス処理の要求をして来ると、プロセッサ３はライトデータをライトバッファ９に格納するよう処理する。そしてそれが完了すると標準バス２０１に書き込み完了を示す信号を出力する。その後、プロセッサ３はライトバッファ９に格納されたデータをフラッシュメモリ１に転送、格納する。この処理を行っている間プロセッサ３は出力ポート信号４２１をアクティブにしてインジケータ４２２を発光させる。そしてライトバッファ９からフラッシュメモリ１への転送が終了したらプロセッサ３は出力ポート４２１をインアクティブにしてインジケータ４２５の発光を停止する。図３５（１）はＩＣカード形状で適用した例で、コネクタとは反対側の側面にインジケータ４２５を取付け、パソコン本体に取り付けた状態でインジケータ４２５の点灯が確認できるようにしている。（２）はノート型パソコンに実際に挿入した様子を示しており、ユーザはインジケータ４２５の点灯、非点灯を確認しながら作業ができる。ただしユーザが点灯を確認しなければならないのは基本的に電源を遮断するときだけである。本実施例によれば、回路構成が比較的単純であり、またインジケータの視認性が良くユーザの誤操作が防げる効果がある。

【００２８】

【発明の効果】本発明によれば消去回数に限りがあるフ

ラッシュメモリを用いた補助記憶装置のデータ管理方式において、特定の論理セクタアドレスの書換えが頻繁に起きても物理的には同一の記憶領域を使用しないため、また消去回数が多くなると消去回数の少ない領域のデータと入替えて消去回数の増加を平均化するため、システムとして寿命が延びる効果がある。またデータメモリ素子数は実際の記憶容量と一致する容量だけでよく冗長のメモリ素子を必要としない。さらにメモリ内にデータ領域だけでなく、情報を保持する領域や、データバッファ領域をもたせることにより、周辺回路の素子数を減らしシステム全体の小型化に大きく貢献する。なおこのフラッシュファイルシステムを搭載する情報機器はＨＤＤを補助記憶装置として動作するのに、実際にはフラッシュメモリを用いた記憶装置である、という構成にすることにより、一般的な情報機器のハードウェアを変更することなしにフラッシュファイルシステムを補助記憶装置とすることができる。またフラッシュメモリの書き換え回数に限界があることにより、記憶装置としての寿命が信頼性に大きく関わるが、使用限界に達したことを認識しユーザに報告する機能を持たせ、記憶装置としての信頼性を向上させることができる。またフラッシュメモリの書き込み速度が遅いために、ホスト側が電源を落されてもフラッシュファイルシステムとしては処理が残っているときには、バックアップバッテリにより処理を継続したり、ホストの電源装置を適宜制御することにより、データの消失を防ぐことができる。またフラッシュメモリの劣化度を消去時間で診断し、劣化が激しくなったらあまり劣化していない領域の格納データを格納することにより、劣化の進行を抑えることができる。フラッシュメモリの劣化は消去時間の長短で認識できるため、正確な劣化の進行度を把握でき、また劣化の段階で記憶すれば劣化度の情報の記憶を比較的少ない記憶容量で行うことができる効果がある。また物理セクタテーブルや論理セクタテーブルを省略して、セクタ番号変換テーブルを備えることにより、システムに応じてテーブル領域を節約することができ、ファイルデータ以外の情報データの格納領域を小さくできる効果がある。またインジケータを補助記憶装置に設けることにより、ホストパソコンの表示画面上のプロンプトだけでは補助記憶装置の動作状態が不明であるため、先述のような電源制御を採用していないシステムの場合には補助記憶装置の動作中に誤って電源を落すのを未然に防ぐことができる。

【図面の簡単な説明】

【図１】本発明における第１の実施例のハードウエア構成図。

【図２】本発明における第１の実施例のフラッシュメモリチップの記憶構成図。

【図３】本発明における第１の実施例の動作を示すメインルーチンのフローチャート

【図４】本発明における第１の実施例の消去管理動作を示す消去管理ルーチンのフローチャート

【図５】本発明における第２の実施例のハードウエア構成図。

【図６】本発明における第２の実施例のフラッシュメモリチップの記憶構成図。

【図７】本発明における第２の実施例の動作を示すメインルーチンのフローチャート

【図８】本発明における第２の実施例のセクタ整理動作を示す整理ルーチンのフローチャート

【図９】本発明における第２の実施例の消去管理動作を示す消去管理ルーチンのフローチャート

【図１０】本発明における第３の実施例のハードウエア構成図。

【図１１】本発明における第３の実施例のフラッシュメモリチップの記憶構成図。

【図１２】本発明における第３の実施例の消去管理動作を示す消去管理ルーチンのフローチャート

【図１３】本発明における第４の発明のハードウエア構成図

【図１４】消去ブロックごとにテーブルを持たせたフラッシュメモリチップの構成図

【図１５】データ領域とは別に情報を格納する領域を持たせたフラッシュメモリチップの構成図

【図１６】データ領域とは別にバッファ領域をもたせたフラッシュメモリチップの構成図

【図１７】本発明における第４の実施例のフラッシュファイルシステムのインターフェース部分のハードウエア構成図。

【図１８】本発明における第４の実施例のフラッシュファイルシステムを補助記憶装置とする情報機器の構成図。

【図１９】本発明における使用限界を報告する実施例の構成図

【図２０】本発明における使用限界を報告する実施例のフローチャート

【図２１】本発明におけるバックアップ電池を備えた実施例の構成図

【図２２】本発明における電源制御信号を備えた実施例の構成図

【図２３】データ領域とは別に情報格納領域をもたせたフラッシュメモリのデータ格納の実施例

【図２４】図１３に対しＥＥＰＲＯＭを省略した実施例の構成図

【図２５】本発明における第５の実施例のハードウエア構成図。

【図２６】本発明における第５の実施例のフラッシュメモリチップの記憶構成図。

【図２７】本発明における第５の実施例の動作を示すメインルーチンのフローチャート

【図２８】本発明における第５の実施例のセクタ整理動

31

作を示す整理ルーチンのフローチャート

【図２９】時間による劣化の管理によりシステムの長寿命化を図る実施例の構成図

【図３０】図２９の実施例における劣化管理コントローラの動作フロー

【図３１】図２９の実施例における記憶システムコントローラの動作フロー

【図３２】消去管理テーブルの記憶容量を節減する実施例の説明図

【図３３】セクタの管理テーブルの記憶容量をを節減するためにアドレス変換テーブルを採用した実施例の構成図

【図３４】補助記憶装置の動作中を示すインジケータを設けた実施例の構成図

【図３５】補助記憶装置の動作中を示すインジケータを設けた実施例の外観図

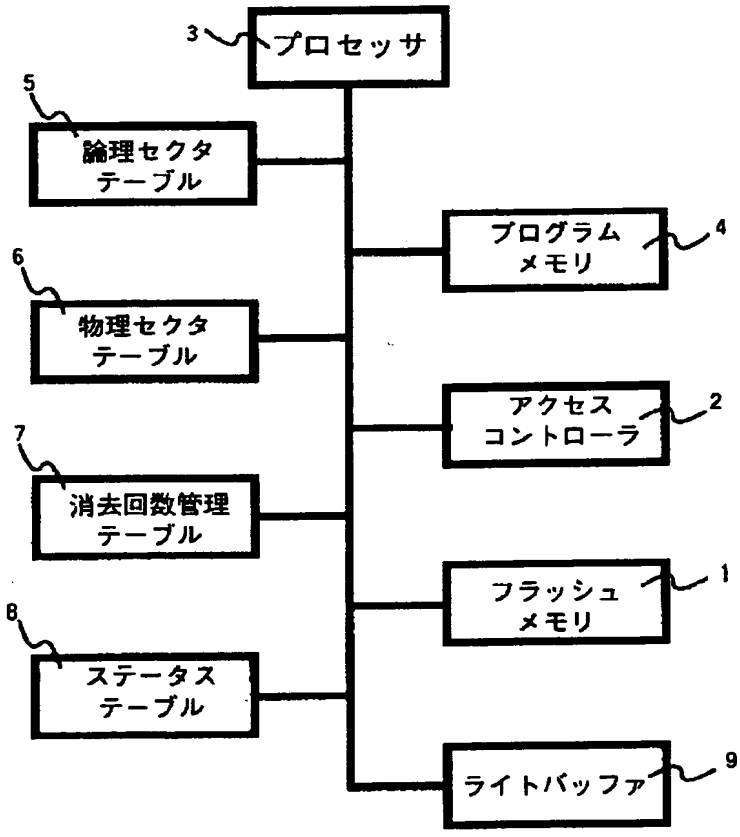
【符号の説明】

32

１…フラッシュメモリ、３…プロセッサ、５…論理セクタテーブル、６…物理セクタテーブル、７…消去回数管理テーブル、８…ステータステーブル、９…ライトバッファ、１３…消去ブロック、４９…書き込みセクタ数テーブル、５１…整理バッファ、５２…物理セクタ、１１１…ワンチップマイコン、１１２…ＲＡＭコア、１１３…ＲＯＭコア、１１４…ＥＥＰＲＯＭ、１１５…ＤＲＡＭまたはＳＲＡＭ、１１６…テーブル領域、１３２…テーブル格納領域、１４２…バッファ領域、１４３…アドレスカウンタ、２０７…Ｉ／Ｆレジスタ群、２２７…ＢＩＯＳＲＯＭ、２３６…Ｉ／Ｆユニット、２４１…エラー報告レジスタ、２５３…バックアップバッテリ、２５９…電源ビジー信号、４０２…消去時間測定タイマ、４０４…劣化度管理テーブル、４１１…アドレス変換テーブル、４２１…出力ポート信号
４２２…インジケータ、４２５…インジケータ（外観）

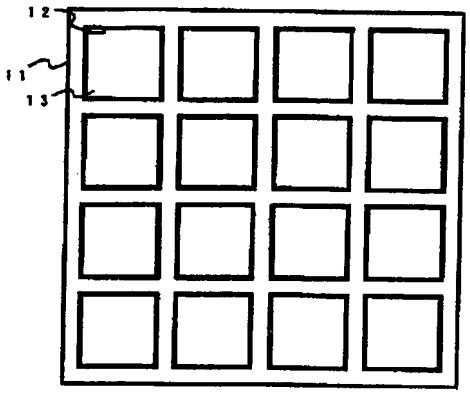
【図１】

図1



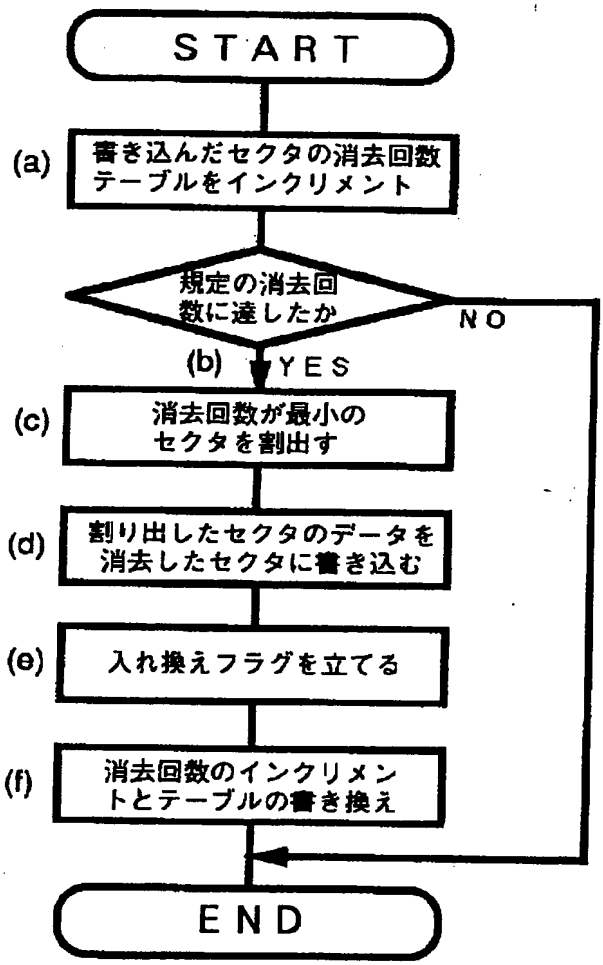
【図2】

図2



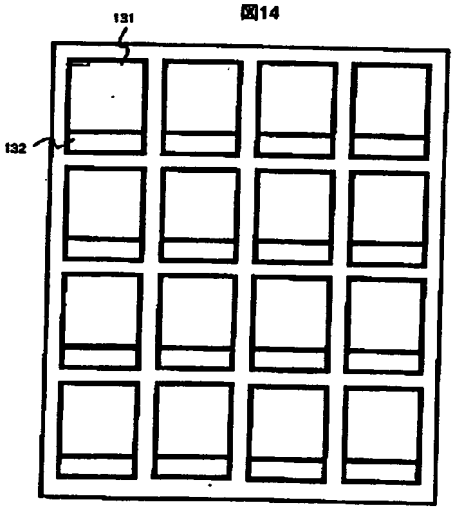
【図4】

図4



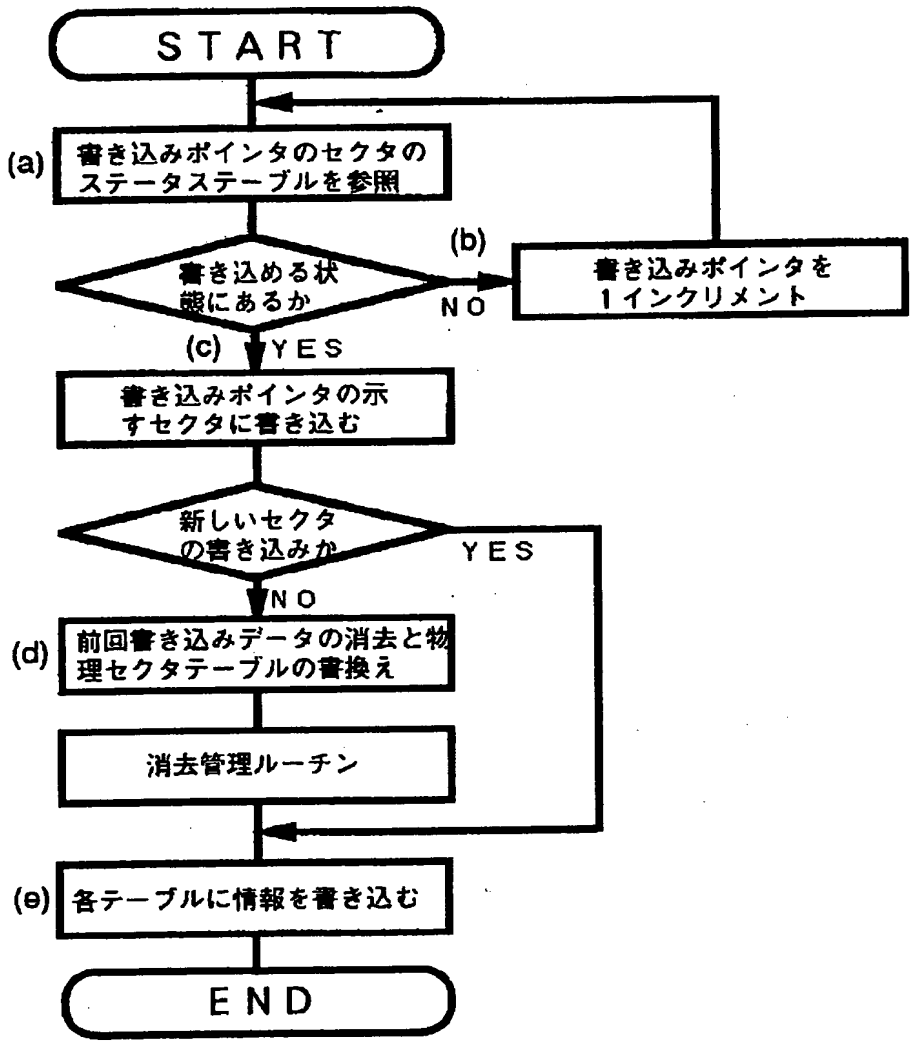
【図14】

図14



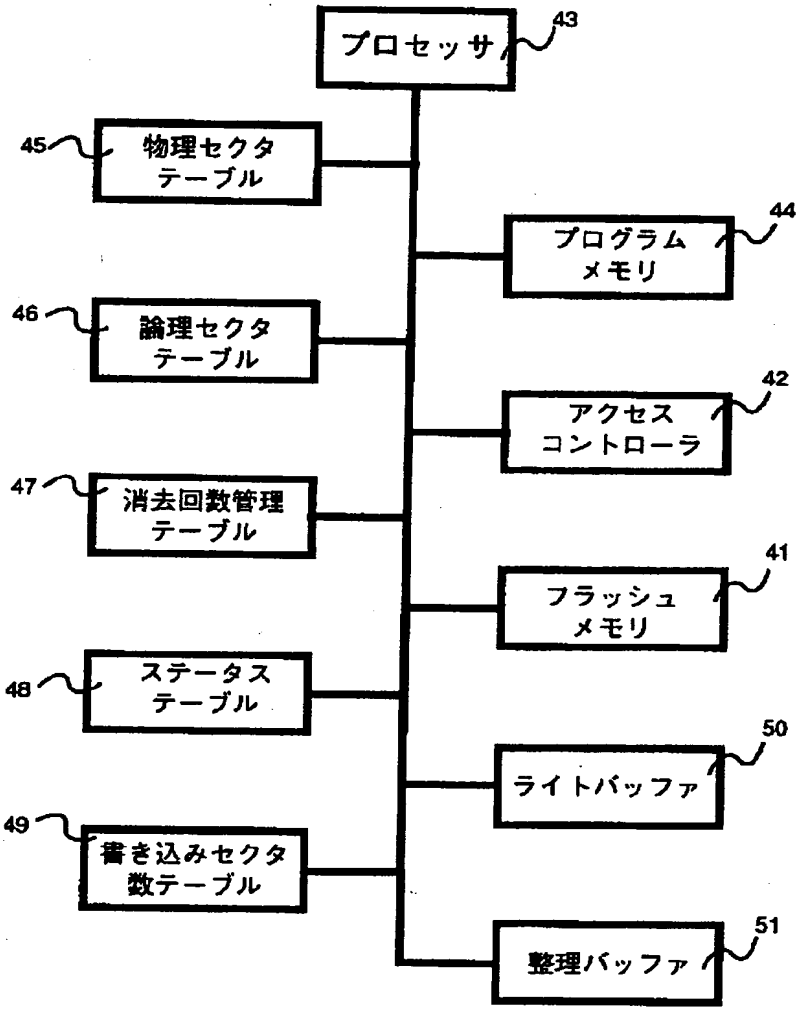
【図3】

図3


START
(a)
書き込みポインタのセクタのステータステーブルを参照
書き込める状態にあるか
(b)
NO
書き込みポインタを1インクリメント
(c)
YES
書き込みポインタの示すセクタに書き込む
新しいセクタの書き込みか
YES
NO
(d)
前回書き込みデータの消去と物理セクタテーブルの書換え
消去管理ルーチン
(e)
各テーブルに情報を書き込む
END

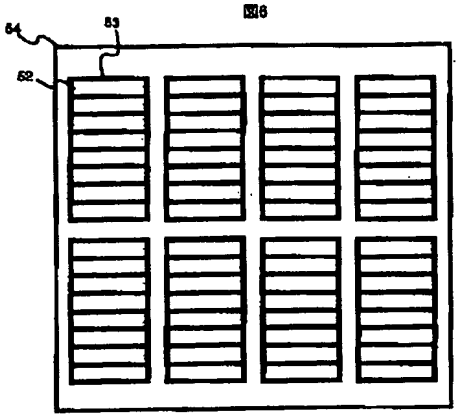
【図5】

図5


プロセッサ
43
物理セクタ
テーブル
45
プログラム
メモリ
44
論理セクタ
テーブル
46
アクセス
コントローラ
42
消去回数管理
テーブル
47
フラッシュ
メモリ
41
ステータス
テーブル
48
ライトバッファ
50
書き込みセクタ
数テーブル
49
整理バッファ
51

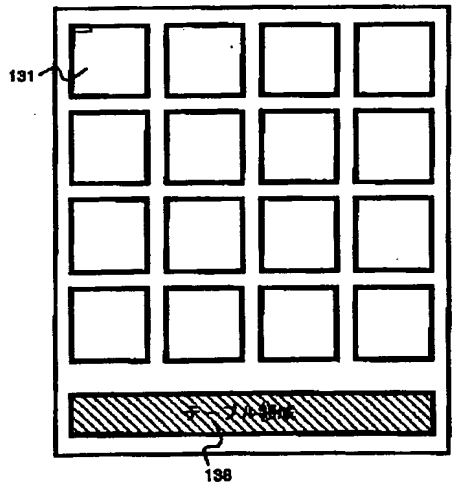
【図6】

図6



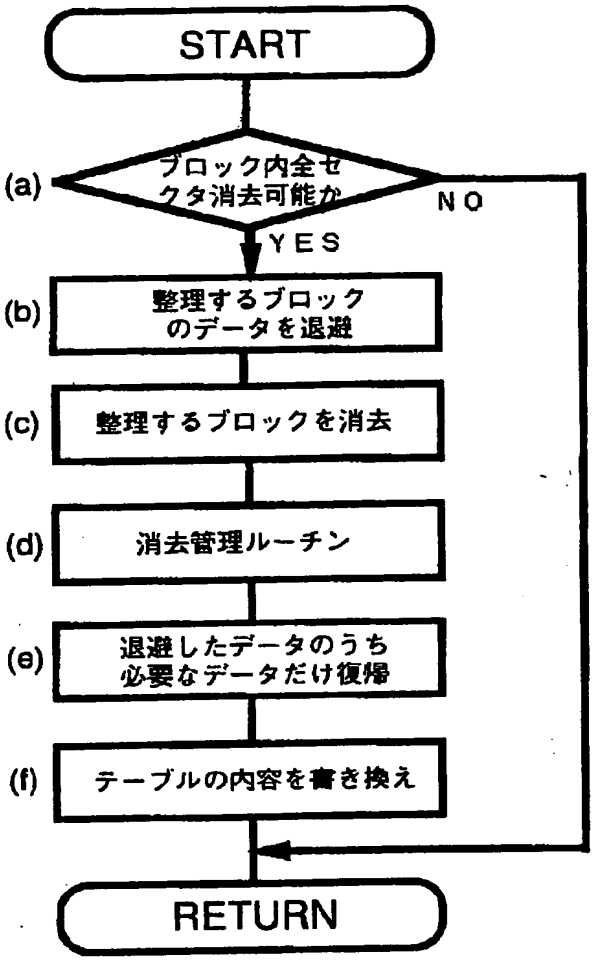
【図15】

図15



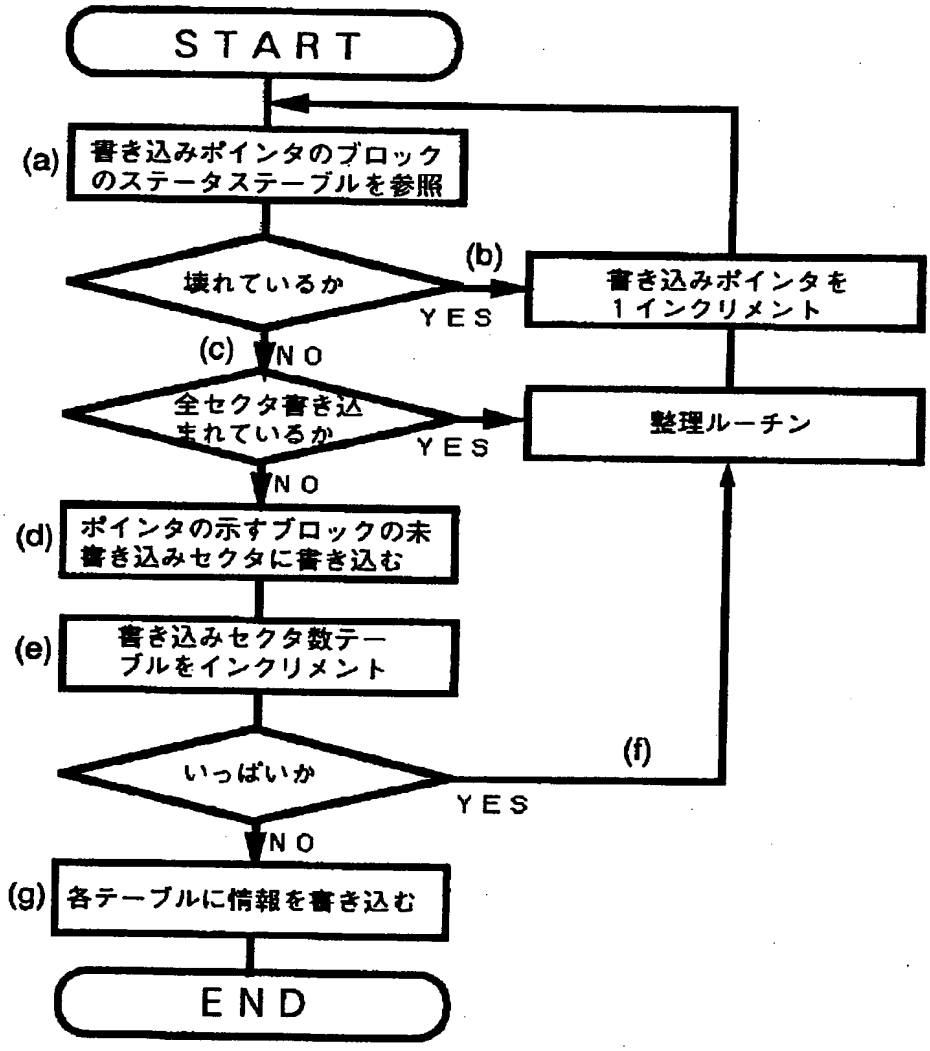
【図8】

図8



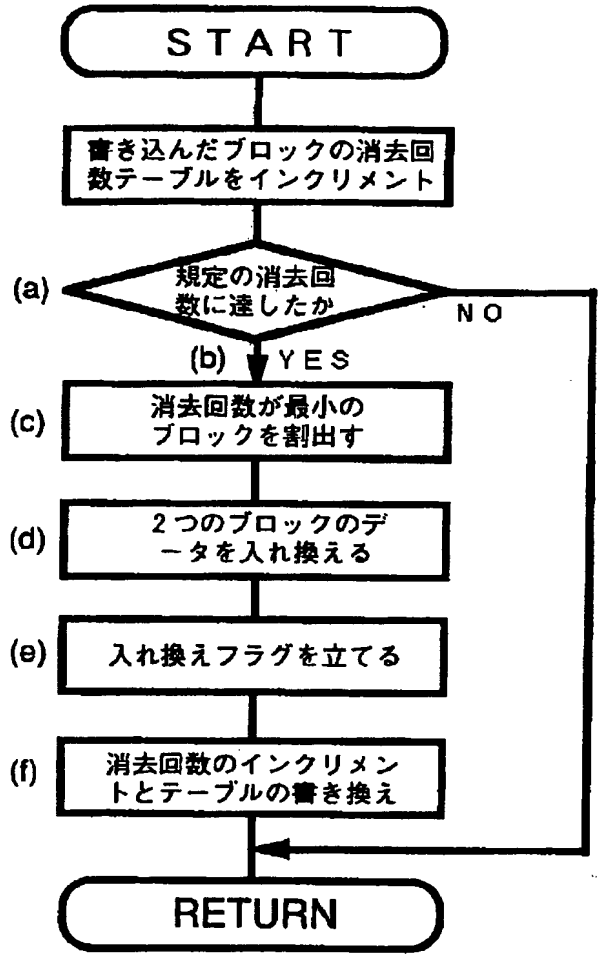
【図7】

図7


START
(a)
書き込みポインタのブロックのステータステーブルを参照
壊れているか
YES
(b)
書き込みポインタを1インクリメント
NO
(c)
全セクタ書き込まれているか
YES
整理ルーチン
NO
(d)
ポインタの示すブロックの未書き込みセクタに書き込む
(e)
書き込みセクタ数テーブルをインクリメント
いっぱいか
(f)
YES
NO
(g)
各テーブルに情報を書き込む
END

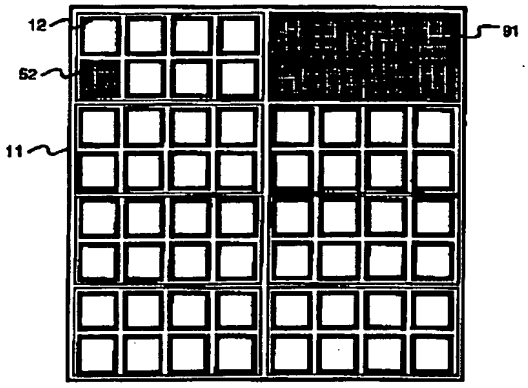
【図9】

図9



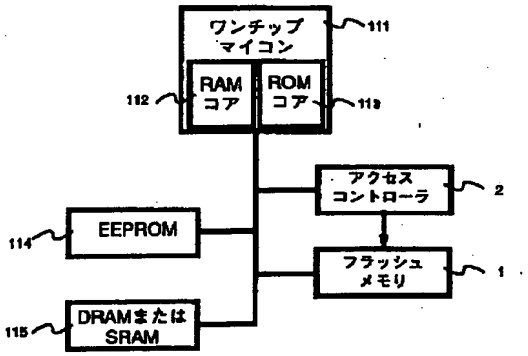
【図10】

図10



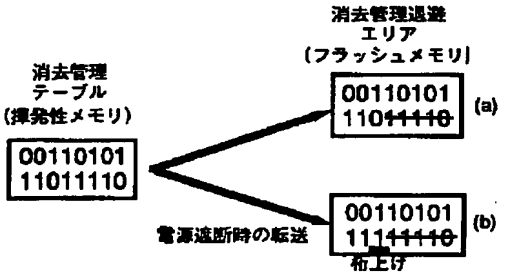
【図13】

図13



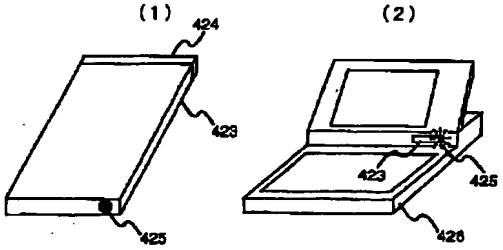
【図32】

図32



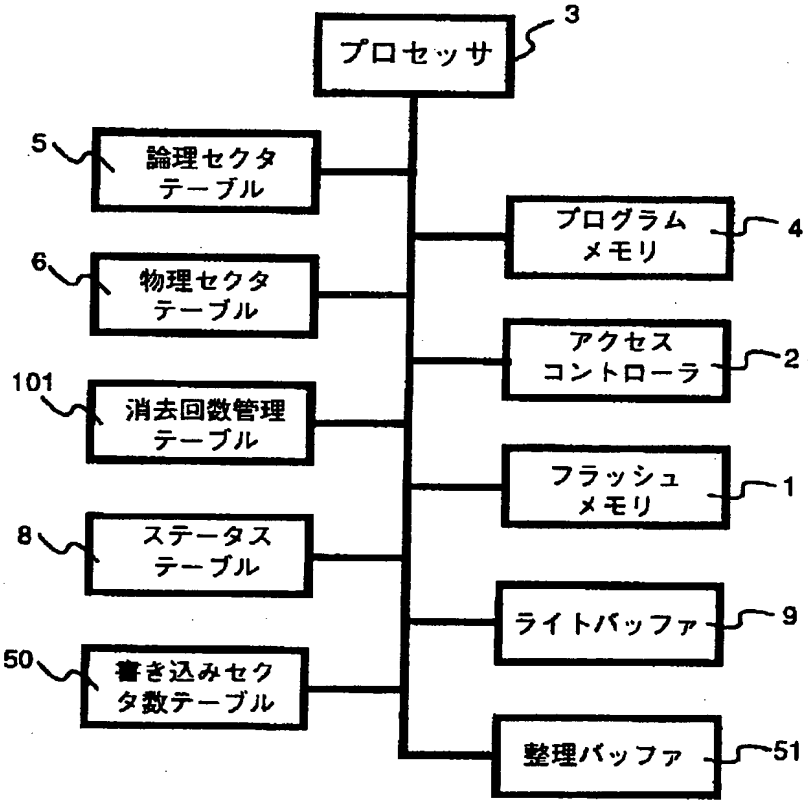
【図35】

図35



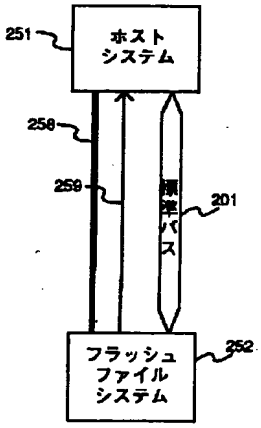
【図11】

図11



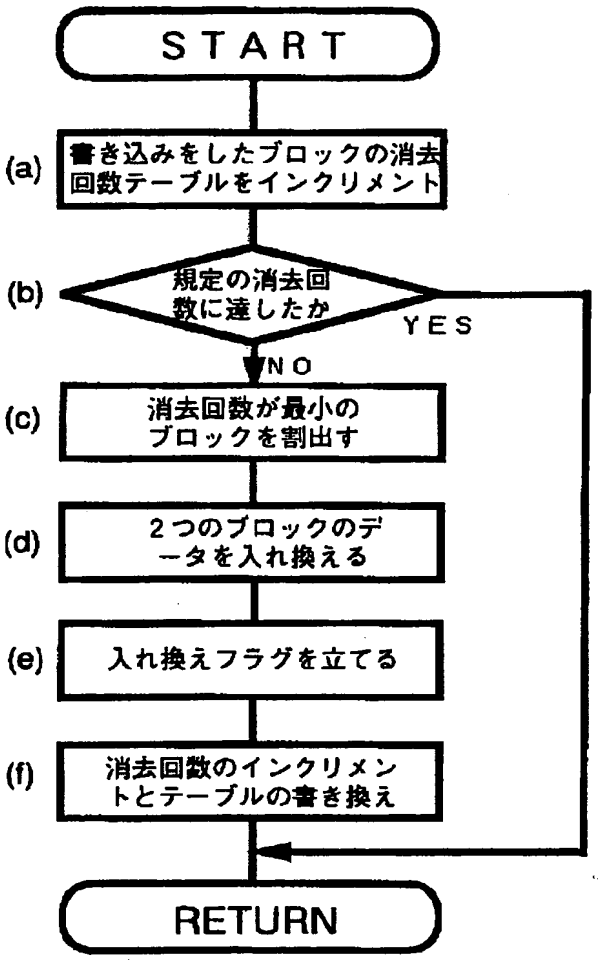
【図22】

図22



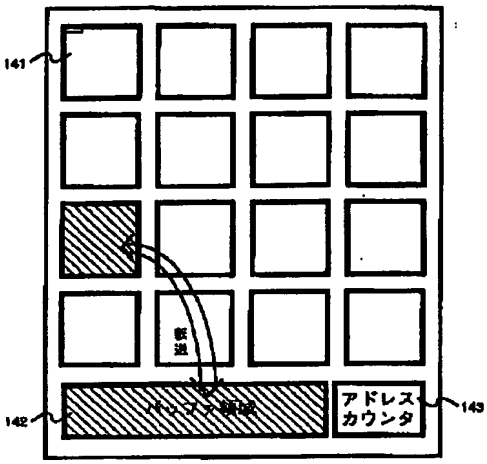
【図12】

図12



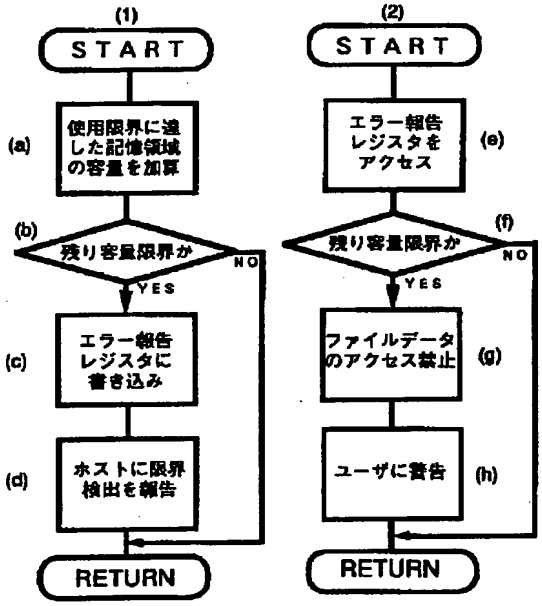
【図16】

図16



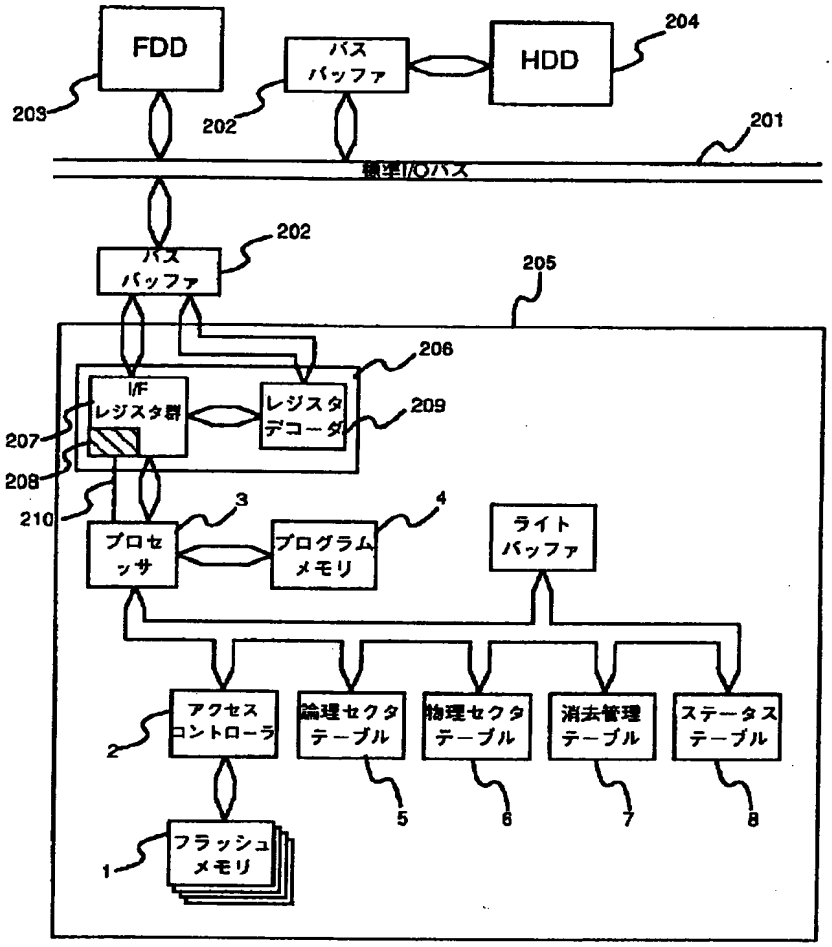
【図20】

図20



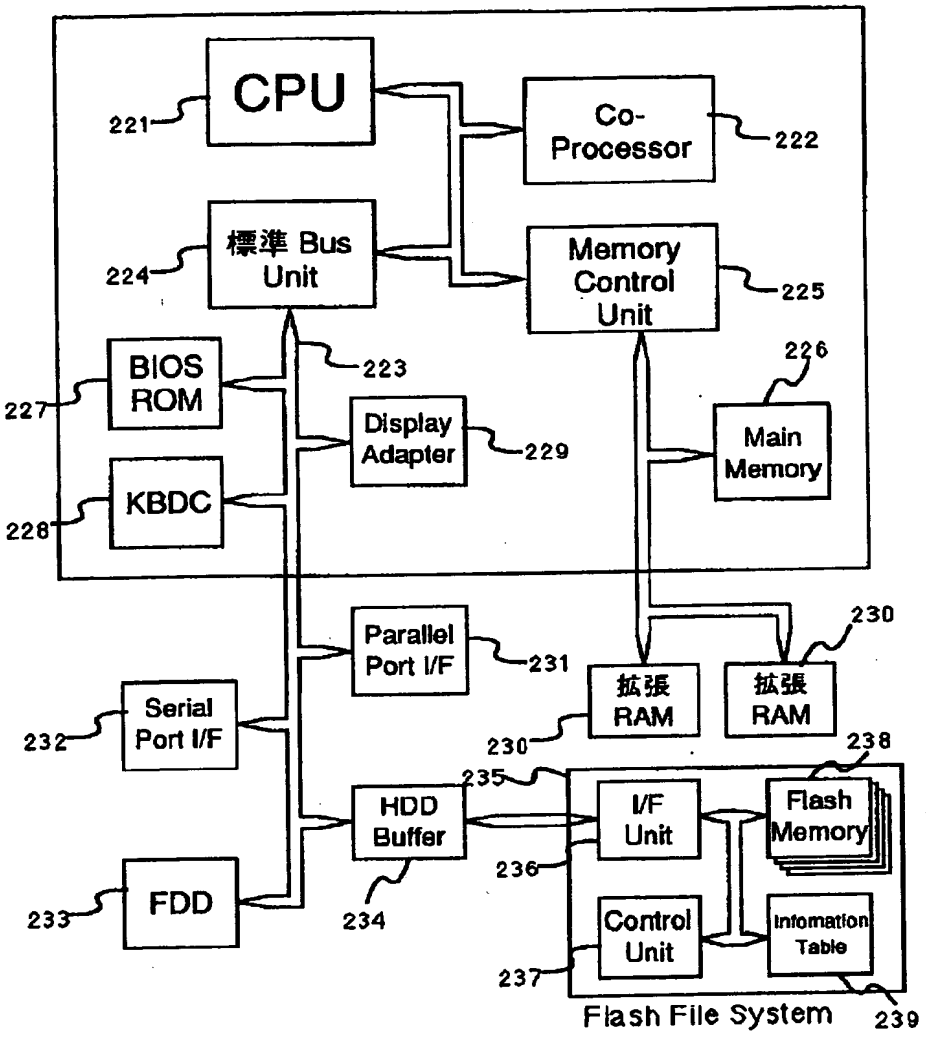
【図17】

図17


FDD
203
バス
バッファ
202
HDD
204
201
標準I/Oバス
202
バス
バッファ
205
206
I/F
レジスタ群
207
208
210
レジスタ
デコーダ
209
3
プロセッサ
4
プログラム
メモリ
ライト
バッファ
2
アクセス
コントローラ
論理セクタ
テーブル
5
物理セクタ
テーブル
6
消去管理
テーブル
7
ステータス
テーブル
8
1
フラッシュ
メモリ

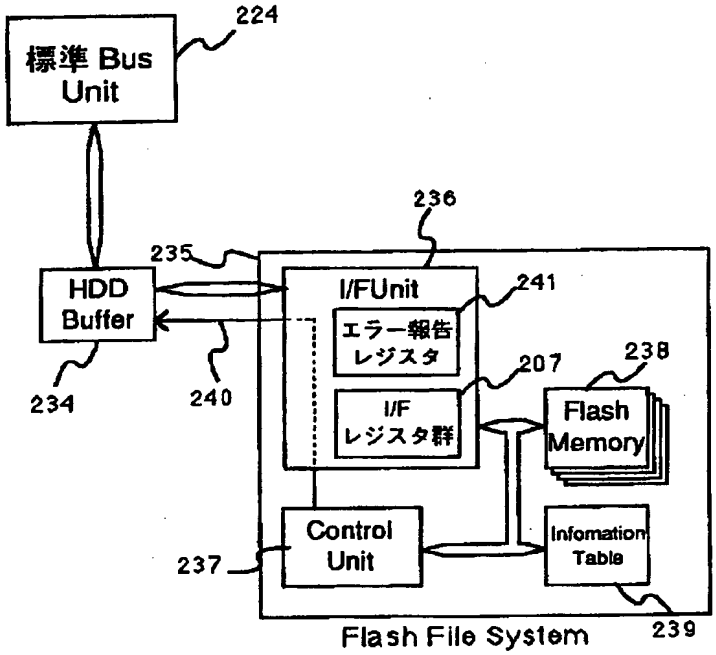
【図18】

図18


CPU
221
Co-
Processor
222
標準 Bus
Unit
224
Memory
Control
Unit
225
223
226
BIOS
ROM
227
Display
Adapter
229
Main
Memory
KBDC
228
Parallel
Port I/F
231
230
拡張
RAM
拡張
RAM
230
Serial
Port I/F
232
238
235
HDD
Buffer
I/F
Unit
Flash
Memory
236
234
Control
Unit
Infomation
Table
FDD
233
237
Flash File System
239

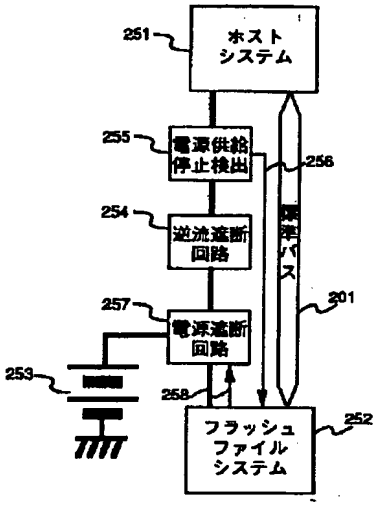
【図19】

図19



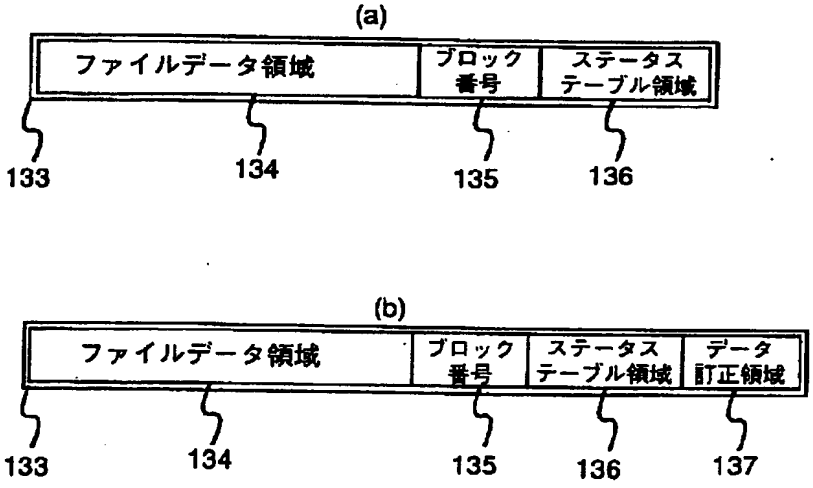
【図21】

図21



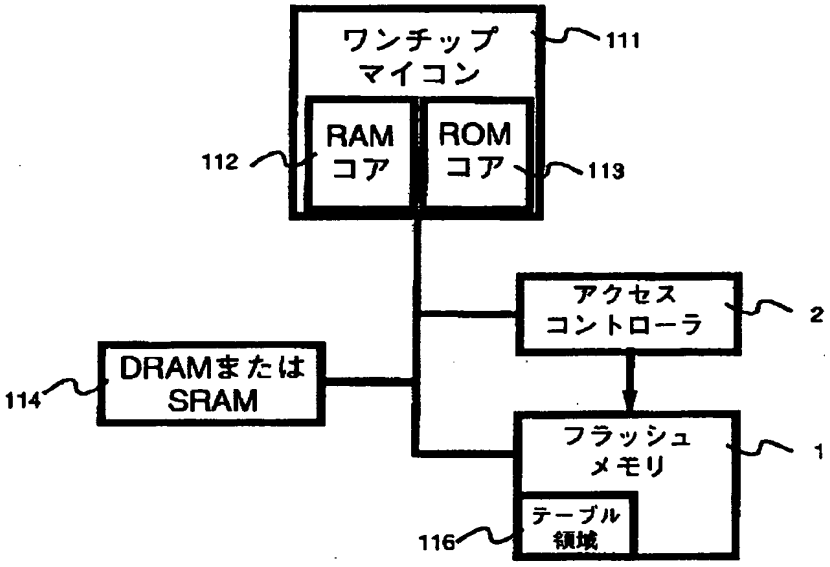
【図23】

図23



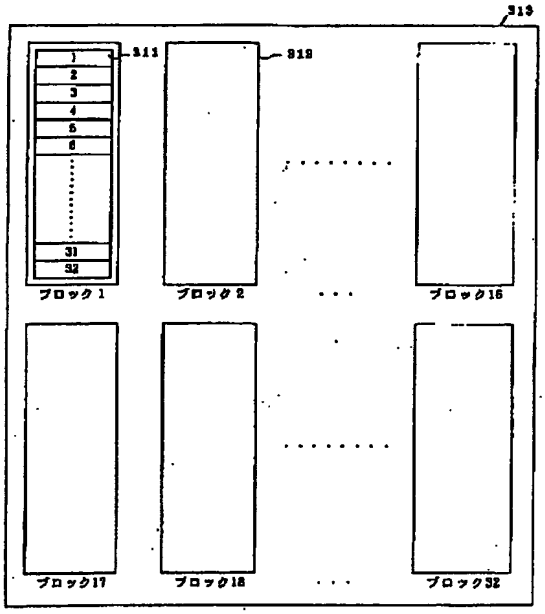
【図24】

図24



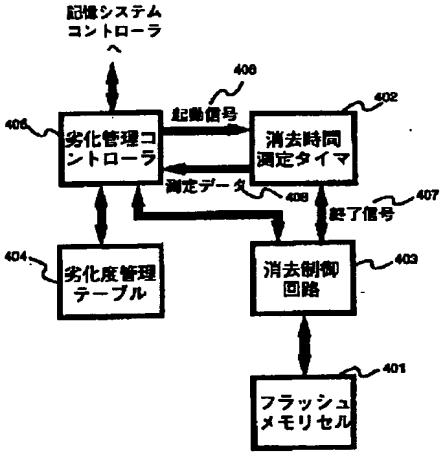
【図26】

図26



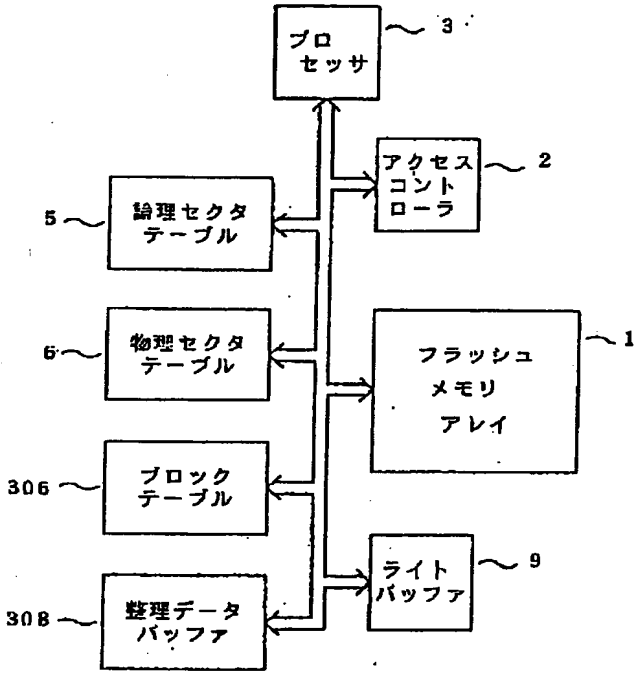
【図29】

図29



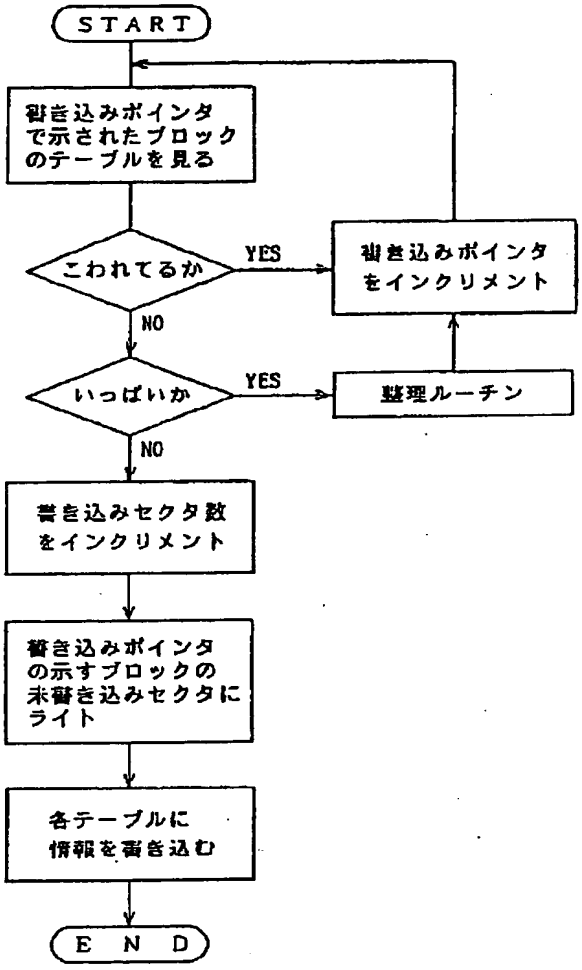
【図25】

図25


プロセッサ
3
アクセスコントローラ
2
論理セクタテーブル
5
物理セクタテーブル
6
フラッシュメモリアレイ
1
ブロックテーブル
306
ライトバッファ
9
整理データバッファ
308

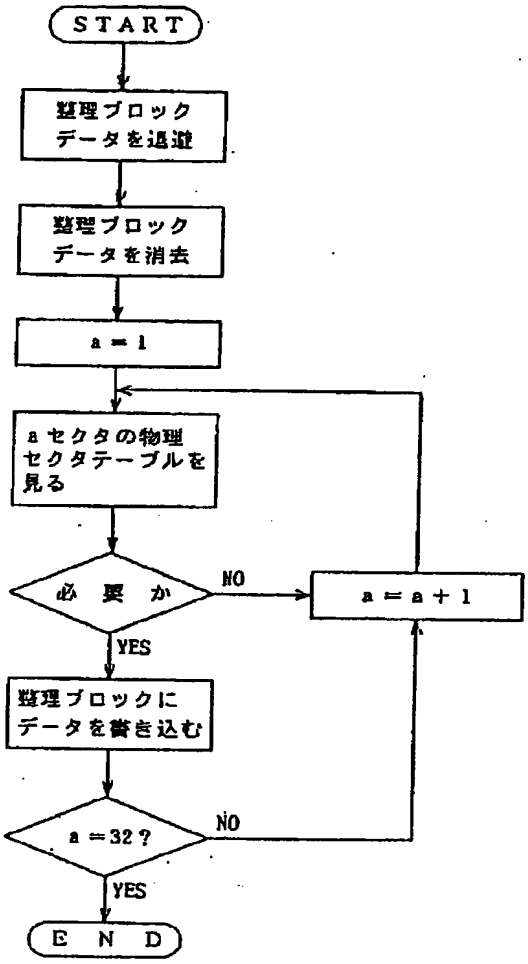
【図27】

図27


START
書き込みポインタで示されたブロックのテーブルを見る
こわれてるか
YES
書き込みポインタをインクリメント
NO
いっぱいか
YES
整理ルーチン
NO
書き込みセクタ数をインクリメント
書き込みポインタの示すブロックの未書き込みセクタにライト
各テーブルに情報を書き込む
E N D

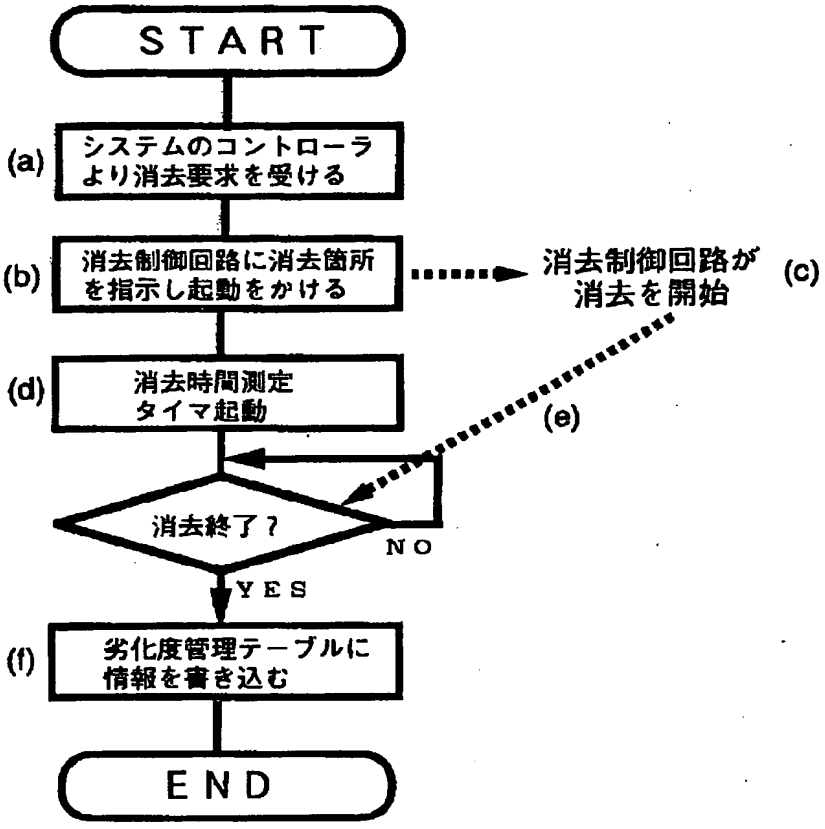
【図28】

図28


START
整理ブロック
データを退避
整理ブロック
データを消去
a＝1
aセクタの物理
セクタテーブルを
見る
必　要　か
NO
a＝a＋1
YES
整理ブロックに
データを書き込む
a＝32？
NO
YES
E　N　D

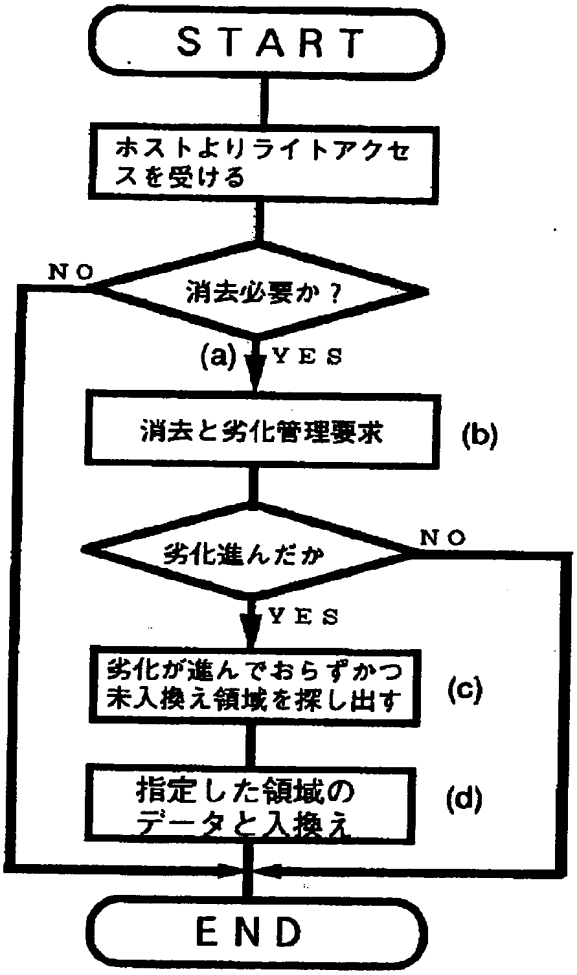
【図30】

図30


START
(a)
システムのコントローラ
より消去要求を受ける
(b)
消去制御回路に消去箇所
を指示し起動をかける
消去制御回路が
消去を開始
(c)
(d)
消去時間測定
タイマ起動
(e)
消去終了？
NO
YES
(f)
劣化度管理テーブルに
情報を書き込む
END

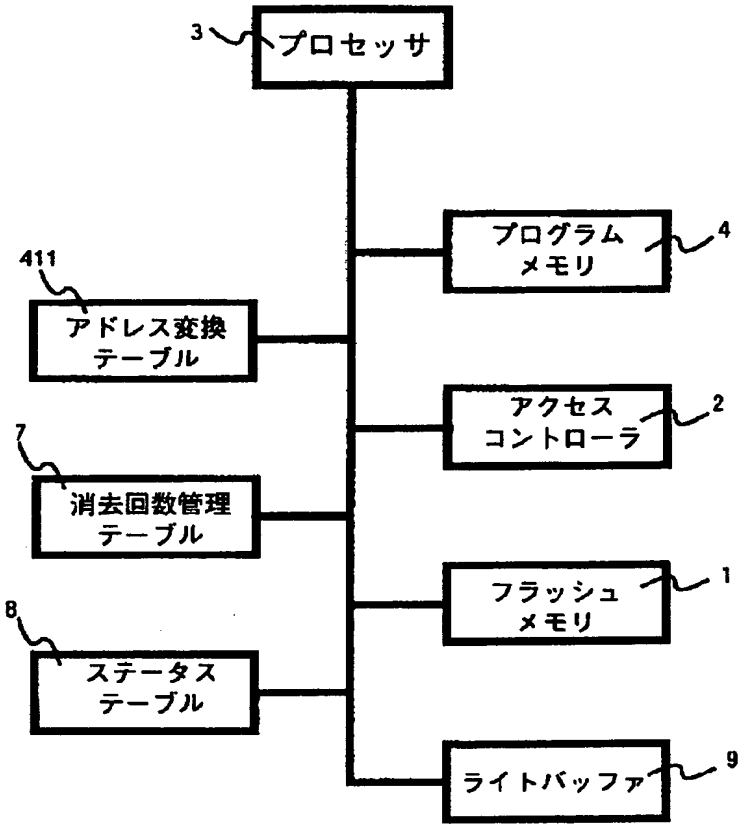
【図31】

図31


START
ホストよりライトアクセスを受ける
NO
消去必要か？
(a)
YES
消去と劣化管理要求
(b)
劣化進んだか
NO
YES
劣化が進んでおらずかつ未入換え領域を探し出す
(c)
指定した領域のデータと入換え
(d)
END

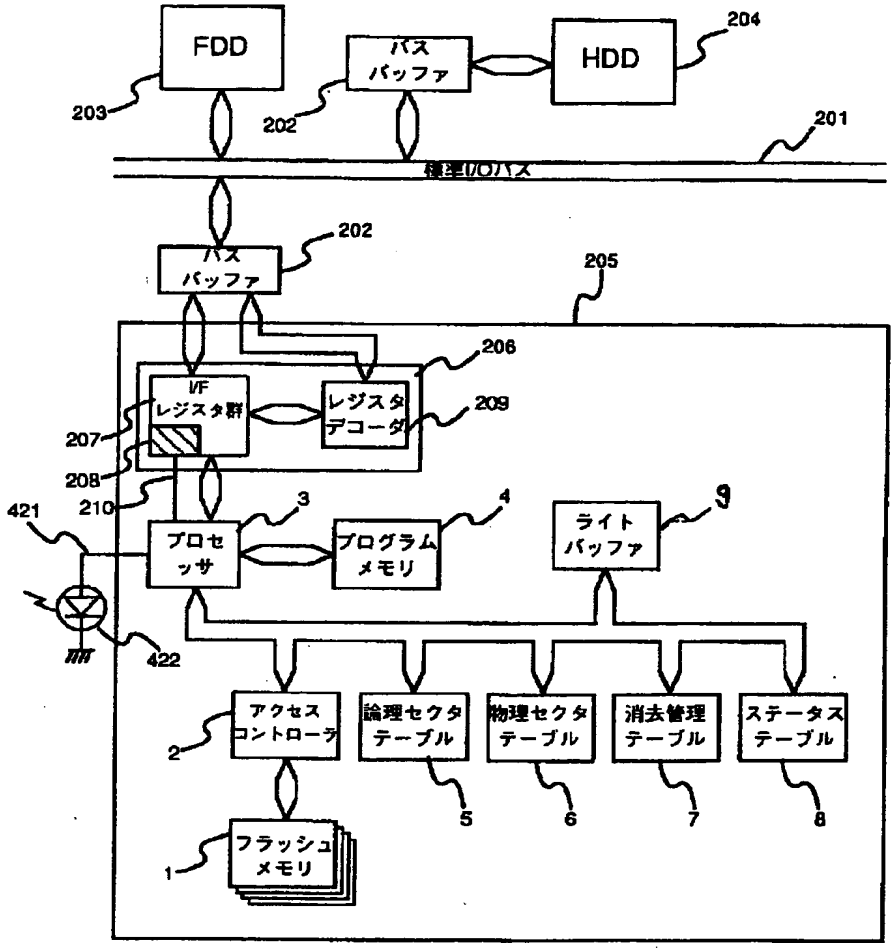
【図33】

図33


3
プロセッサ
プログラム
メモリ
4
411
アドレス変換
テーブル
アクセス
コントローラ
2
7
消去回数管理
テーブル
フラッシュ
メモリ
1
8
ステータス
テーブル
ライトバッファ
9

【図３４】

図３４



フロントページの続き

(72)発明者　柿　健一
神奈川県横浜市戸塚区吉田町292番地株式会社日立製作所マイクロエレクトロニクス機器開発研究所内
(72)発明者　和出　武史
東京都小平市上水本町五丁目20番１号株式会社日立製作所半導体設計開発センタ内
(72)発明者　古野　毅
東京都小平市上水本町五丁目20番１号株式会社日立製作所半導体設計開発センタ内
(72)発明者　戸塚　隆
東京都小平市上水本町五丁目20番１号株式会社日立製作所半導体設計開発センタ内